



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

GRADUAL SYSTEM F

TESIS PARA OPTAR AL GRADO DE
DOCTORA EN CIENCIAS, MENCIÓN COMPUTACIÓN

ELIZABETH LABRADA DENIZ

PROFESOR GUÍA:
ÉRIC TANTER

PROFESOR CO-GUÍA:
MATÍAS TORO IPINZA

MIEMBROS DE LA COMISIÓN:
DOMINIQUE DEVRIESE
FEDERICO OLMEDO BERON
ISMAEL FIGUEROA PALET

SANTIAGO DE CHILE
2022

Resumen

SISTEMA F GRADUAL

El polimorfismo paramétrico es un mecanismo de abstracción ampliamente utilizado en los lenguajes de programación, que permite la definición genérica de funciones y tipos, y proporciona reutilización de código e independencia de representación. Por otra parte, la tipificación gradual permite una transición paulatina entre la comprobación de tipos estática y dinámica basada en anotaciones de tipos controladas por el programador, integrando las ventajas de ambos mundos. Llevar los beneficios del tipado gradual a un lenguaje con polimorfismo paramétrico como el Sistema F, preservando al mismo tiempo la parametricidad relacional, ha resultado ser un gran reto. La dificultad observada en trabajos anteriores se basa en una fuerte tensión entre las dos propiedades deseables de un lenguaje paramétrico gradual, la parametricidad, y la gradualidad, considerando que las funciones pueden intentar utilizar la tipificación gradual para eludir la parametricidad. Los primeros intentos se formularon hace más de una década, y se han propuesto además varios diseños con sintaxis, comportamientos y propiedades variables. Se ha demostrado que, sin tomar precauciones, la parametricidad podría ser violada en un lenguaje gradual. Por esta razón, todos los lenguajes polimórficos graduales propuestos han utilizado alguna forma de mecanismo de sellado/desellado. La idea detrás del sellado/desellado es que un valor sellado con una clave sólo puede ser manipulado por una expresión de desellado que conozca la clave; de lo contrario, se lanza un error.

Esta tesis contribuye a la adopción del polimorfismo paramétrico en un lenguaje gradual, preservando la sintaxis del Sistema F y satisfaciendo la gradualidad y parametricidad al mismo tiempo. Exploramos en profundidad la integración de estas dos características derivando dos lenguajes, GSF y Funky. GSF es un lenguaje fuente paramétrico gradual que viola la gradualidad en algunos escenarios. No obstante, establecemos una propiedad más débil que nos permite refutar varias afirmaciones sobre teoremas gratis graduales, aclarando el tipo de razonamiento que admite la parametricidad gradual. Funky es un lenguaje intermedio que va más allá, satisfaciendo plenamente la parametricidad y la gradualidad, y permitiendo la incrustación de diferentes lenguajes fuentes paramétricos graduales. Basándonos en la observación de que la tensión entre gradualidad y parametricidad proviene de la decisión temprana de sellar valores basada en la información de tipo, proponemos el sellado plausible como un nuevo mecanismo de lenguaje intermedio que permite posponer tales decisiones al tiempo de ejecución. Funky soporta el sellado plausible en un entorno simplificado en el que el polimorfismo se restringe a las instancias con tipos base y variables. Aunque los resultados presentados tienen algunas limitaciones, proponen una nueva forma de reconciliar la parametricidad y la gradualidad en un lenguaje con la sintaxis del Sistema F. Además, algunas de nuestras técnicas novedosas son potencialmente reutilizables en otros entornos.

Abstract

Parametric polymorphism is an abstraction mechanism widely used in programming languages, enabling the generic definition of functions and types and providing code reusability and representation independence. On the other hand, gradual typing supports the smooth transition between static and dynamic type checking based on programmer-controlled type annotations, integrating the benefits of both worlds. Bringing the advantages of gradual typing to a language with parametric polymorphism like System F while preserving relational parametricity has proven extremely challenging. The difficulty observed in these efforts is a strong tension between the two desirable properties of a gradual parametric language, parametricity, and graduality, considering that functions may attempt to use gradual typing to bypass parametricity. First attempts were formulated more than a decade ago, and several designs have been proposed with varying syntax, behavior, and properties. It has been shown that without further precaution, parametricity might be violated in a gradual language. For this reason, all gradual polymorphic languages proposed have used some form of sealing/unsealing mechanism. The idea behind sealing/unsealing is that a value sealed with a key can only be manipulated by an unseal expression that knows the key; otherwise, an error is thrown.

This thesis contributes to adopting parametric polymorphism in a gradual language, preserving the syntax of System F and satisfying graduality and parametricity simultaneously. We deeply explore the integration of these two features by deriving two languages, GSF, and Funky, thereby developing the metatheory of gradual parametricity. GSF is a gradual parametric source language that violates graduality in some specific scenarios. We nevertheless establish a weaker property than graduality that allows us to disprove several claims about gradual free theorems, clarifying the kind of reasoning supported by gradual parametricity. Funky is an intermediate language that goes further, fully satisfying parametricity and graduality, and allowing the embedding of different gradual parametric source languages. Based on the observation that the tension between graduality and parametricity comes from the early commitment to seal values based on type information, we propose plausible sealing as a new intermediate language mechanism that allows postponing such decisions to runtime. Funky supports plausible sealing in a simplified setting where polymorphism is restricted to instantiations with base and variable types. Although the results presented here still have some constraints, they propose a new way of reconciling parametricity and graduality in a language with the System F syntax. Furthermore, some of our novel techniques are potentially reusable in other settings.

Table of Content

1	Introduction	1
2	Gradual Parametricity: Background and Basics	6
2.1	Parametric Polymorphism	6
2.2	Gradual Typing	8
2.3	Gradual Parametricity in a Nutshell	11
2.4	Gradual Parametricity: Challenges	12
I	Gradual Parametricity, Revisited	18
3	Gradual System F	19
3.1	GSF, Informally	20
3.1.1	Design Principles	20
3.1.2	Properties	20
3.1.3	GSF in Action	21
3.2	Preliminary: The static language SF	22
3.3	GSF: Statics	25
3.3.1	Syntax and Syntactic Meaning of Gradual Types	25
3.3.2	Lifting the Static Semantics	27
3.3.3	Static Properties of GSF	29
3.4	GSF: Evidence-Based Dynamics	30
3.4.1	Background: Evidence-Based Semantics for Gradual Languages	30

3.4.2	Reduction for GSF	31
3.4.3	Elaborating GSF to GSF_ε	35
3.5	GSF: Evidence for Gradual Parametricity	36
3.5.1	Simple Evidence, and Why It Fails	36
3.5.2	Refining Evidence	39
3.5.3	Basic Properties of GSF Evaluation	45
3.6	GSF and the Dynamic Gradual Guarantee	46
3.6.1	Violation of the Dynamic Gradual Guarantee in GSF	47
3.6.2	Towards a Weak Dynamic Gradual Guarantee for GSF	48
3.6.3	Weak Dynamic Gradual Guarantee for GSF	50
3.6.4	Syntactic Strict Precision for GSF	54
3.7	Gradual Parametricity for GSF	57
3.7.1	On Gradual Parametricities	57
3.7.2	Gradual Parametricity in GSF	63
3.7.3	Parametricity vs. the DGG in GSF	68
3.7.4	Gradual Free Theorems in GSF	70
3.8	Related Work	71
3.9	Conclusion	72
4	Embedding Dynamic Sealing in GSF	73
4.1	Overview	73
4.2	Embedding a Dynamically-Typed Language in GSF	74
4.3	The Cryptographic Lambda Calculus λ_{seal}	75
4.4	Embedding λ_{seal} in GSF	76
4.5	Semantic Preservation of the λ_{seal} Embedding in GSF	78
4.6	Related Work	82
4.7	Conclusion	82

5	Gradual Existential Types in GSF	83
5.1	Existential Types in a Nutshell	83
5.2	Existential Types: Primitive or Encoded?	84
5.3	Gradual Existential Types in GSF^\exists	85
5.4	Semantics of GSF^\exists	87
5.4.1	The Static Language SF^\exists	87
5.4.2	GSF^\exists : Statics	89
5.4.3	GSF^\exists : Dynamics	89
5.5	Properties of GSF^\exists	92
5.5.1	Static Properties	92
5.5.2	Dynamic Gradual Guarantees	93
5.5.3	Parametricity	93
5.5.4	Representation Independence	95
5.6	Related Work	96
5.7	Conclusion	97
II	Plausible Sealing for Gradual Parametricity	98
6	The Gradual Language for Plausible Sealing $\text{F}_\varepsilon^?$	101
6.1	Overview of $\text{F}_\varepsilon^?$	101
6.1.1	Evidence for Plausible Sealing	102
6.1.2	Comparing Plausible Sealing and Prior Approaches	103
6.2	The Evidence-Based Language $\text{F}_\varepsilon^?$	104
6.2.1	Syntax and static semantics	104
6.2.2	Dynamic semantics	106
6.3	Evidence for Plausible Sealing in $\text{F}_\varepsilon^?$	107
6.3.1	Proof-relevant precision	107
6.3.2	Evidence and consistent transitivity	109

6.4	$F_\epsilon^?$: Gradual Parametricity	113
6.5	$F_\epsilon^?$: Gradual Guarantees	116
6.5.1	Evidence Precision	116
6.5.2	Term precision	118
6.6	Related Work	119
6.7	Conclusion	120
7	The Gradual Source Language $F^?$	121
7.1	$F^?$: Statics	121
7.2	$F^?$: Elaboration to $F_\epsilon^?$	124
7.3	Source-level graduality	130
7.4	Source-level parametric reasoning	130
7.5	Discussion and Related Work	133
7.6	Conclusion	134
8	Conclusion	135
	Bibliography	137
	ANNEXES	143
	Annex A Gradual Parametricity, Revisited	144
A.1	SF: Well-formedness	144
A.2	GSF: Statics	145
A.2.1	Syntax and Syntactic Meaning of Gradual Types	145
A.2.2	Lifting the Static Semantics	146
A.2.3	Well-formedness	149
A.2.4	Static Properties	150
A.3	GSF: Dynamics	156
A.3.1	Evidence Type Precision	156

A.3.2	Initial Evidence	156
A.3.3	Consistent Transitivity	156
A.3.4	GSF_ε : Dynamic Semantics	156
A.3.5	Translation from GSF to GSF_ε	158
A.4	GSF: Properties	161
A.4.1	Type Safety	161
A.4.2	Static Terms Do Not Fail	165
A.5	GSF and The Dynamic Gradual Guarantee	169
A.5.1	Evidence Type Precision	169
A.5.2	Monotonicity of Evidence Transitivity and Instantiation	169
A.5.3	Weak Dynamic Gradual Guarantee for GSF	178
A.5.4	Syntactic Strict Precision for GSF	188
A.6	GSF: Parametricity	194
A.6.1	Auxiliary Definitions	194
A.6.2	Fundamental Property	194
A.6.3	Contextual Equivalence	234
A.7	Parametricity vs. the DGG in GSF	236
A.8	A Cheap Theorem in GSF	237
A.9	Embedding Dynamic Sealing in GSF	239
A.10	Gradual Existential Types in GSF	265
A.10.1	Existential types: primitive or encoded?	265
A.10.2	Translation from GSF^\exists to $\text{GSF}_\varepsilon^\exists$	266
A.10.3	Properties of GSF^\exists	266
A.10.4	GSF^\exists : Parametricity	267
A.10.5	A Weak Dynamic Gradual Guarantee for GSF^\exists	284

Annex B Plausible sealing **295**

B.1	The Evidence-Based Language $F_\varepsilon^?$	295
-----	---	-----

B.2	Evidence in $F_\epsilon^?$	300
B.2.1	Proof Relevant Gradual Type Precision	300
B.2.2	Pullback Operator	306
B.2.3	Associativity of consistent transitivity	315
B.2.4	Evidence operations	323
B.3	$F_\epsilon^?$: Gradual Parametricity	325
B.4	$F_\epsilon^?$: Gradual Guarantees	392
B.5	The Static Language System F_1	414
B.6	The Gradual Source Language $F^?$	414
B.6.1	Lifting the Static Semantics	414
B.6.2	Typing judgement	429
B.6.3	Elaborating $F^?$ to $F_\epsilon^?$	429
B.6.4	Gradual Guarantees	433
B.6.5	Parametricity	462
B.6.6	Source-level parametric reasoning	463
B.7	Examples	469

Chapter 1

Introduction

Parametric polymorphism is an abstraction mechanism widely used in programming languages, enabling the generic definition of functions and types and providing code reusability and representation independence. System F (Reynolds, 1974; Girard, 1972) is the standard language to formalize this notion of parametric polymorphism. *Relational parametricity* (Reynolds, 1983) is a rich semantic property of parametric polymorphism, which stipulates that the behavior of polymorphic functions must be independent of the specific types they are instantiated with. For instance, the behavior of a function f of type $\forall X. X \rightarrow X$ should not depend on the type X it is instantiated with and, consequently, should treat the argument of type X opaquely. Hence, f [Int] 42 should never return a different value than 42.

Most languages today have either static typing or dynamic typing, and both have complementary strengths. Static typing provides early error detection and efficient code generation, but it can be cumbersome to adapt. Conversely, dynamic typing enables fast adaptation to change requirements and flexibility, at the cost of having runtime checks and errors, making the execution slower. There are many approaches for integrating static and dynamic type checking (Cartwright and Fagan, 1991; Abadi et al., 1991; Matthews and Findler, 2007; Tobin-Hochstadt and Felleisen, 2006; Bierman et al., 2010). In particular, gradual typing supports the smooth integration of static and dynamic type checking by introducing the notion of *imprecision* at the level of types, which induces a notion of *consistency* between plausibly equal types (Siek and Taha, 2006). Imprecise types can range from fully-precise static types to the fully-imprecise unknown type (hereafter written $?$), with partially specified types in between, such as $\text{Int} \rightarrow ?$. Type $\text{Int} \rightarrow ?$ is interpreted as the type of any function receiving an Int as an argument and returning an expression of any type, such as Int , Bool , or another function.

A gradual type checker treats imprecision optimistically, and the runtime of the gradual language detects when optimistic static assumptions are invalid. Such detection is usually achieved by compiling to an internal language with explicit casts, called a cast calculus. For instance, the source gradual term $\lambda x : ?.x + 1$ would typically be elaborated to the target term $\lambda x : ?.\langle \text{Int} \Leftarrow ? \rangle x + 1$, where the cast $\langle \text{Int} \Leftarrow ? \rangle$ ensures that the argument given at runtime is indeed an Int value, otherwise an error is raised.

In addition to being type safe, a gradually-typed language is expected to satisfy a number

of properties that characterize the static-to-dynamic checking spectrum supported by the language (Siek et al., 2015a; New and Ahmed, 2018). A particularly challenging one is the *dynamic gradual guarantee* (DGG) (Siek et al., 2015a), also called *graduality* (New and Ahmed, 2018). Informally, graduality is a monotonicity property of reduction with respect to precision: introducing imprecision in a program ought not to change its behavior. For instance, the function $\lambda x : ?. x + 1$ of type $? \rightarrow \mathbf{Int}$ should be transparently usable in place of $\lambda x : \mathbf{Int}. x + 1$. Although the DGG and graduality have the same purpose: characterizing the static-to-dynamic transition of a gradual language, they are formalized differently. While the DGG by Siek et al. (2015a) is based on a syntactic notion of precision, graduality by New and Ahmed (2018) is based on a more semantic notion. For example, graduality by New and Ahmed (2018) requires embedding-projection pairs (going to a less precise type and back is the identity): for any term t of type G , and given $G \sqsubseteq G'$, then $t :: G' :: G$ should be observationally equivalent to t (we write $t :: G$ for type ascriptions). In this thesis, the terms DGG and graduality are used interchangeably to refer to the same concept.

Since its early formulation in a simple functional language (Siek and Taha, 2006), gradual typing has been explored in a number of increasingly challenging settings such as subtyping (Siek and Taha, 2007; Garcia et al., 2016), references (Herman et al., 2010; Siek et al., 2015b), effects (Bañados Schwerter et al., 2014, 2016), ownership (Sergey and Clarke, 2012), tpestates (Wolff et al., 2011; Garcia et al., 2014), information-flow typing (Disney and Flanagan, 2011; Fennell and Thiemann, 2013; Toro et al., 2018), session types (Igarashi et al., 2017b), refinements (Lehmann and Tanter, 2017), dependent types (Eremondi et al., 2019), set-theoretic types (Castagna and Lanvin, 2017), Hoare logic (Bader et al., 2018), separation logic (Wise et al., 2020), and, most relevant to this work, parametric polymorphism (Ahmed et al., 2011, 2017; Ina and Igarashi, 2011; Igarashi et al., 2017a; Xie et al., 2018; New et al., 2020).

It turns that out that integrating parametric polymorphism and gradual typing into a language while preserving parametricity and graduality is extremely challenging (Matthews and Ahmed, 2008; Ahmed et al., 2009b, 2017; New et al., 2020; Igarashi et al., 2017a). The difficulty observed in these efforts is a strong tension between the two desirable properties, considering that functions may attempt to use gradual typing to bypass parametricity. For example, the following function of type $\forall X. X \rightarrow X$ should not be allowed to treat the value x as an integer, even when X happens to be instantiated to \mathbf{Int} :

$$(\Lambda X. \lambda x X. ((x :: ?) + 1) :: X) [\mathbf{Int}] 42$$

Observe that if this program reduces to 43, a violation of parametricity will occur because of the addition to the parametric argument. To prevent this application from reducing to $((42 :: ?) + 1) :: \mathbf{Int}$, gradual polymorphic languages have generally relied on a form of *dynamic sealing* (Morris, 1973; Matthews and Ahmed, 2008). Essentially, the function $(\Lambda X. \lambda x X. ((x :: ?) + 1) :: X)$ is not applied to type \mathbf{Int} and value 42; instead, the language generates at runtime a fresh seal α and applies the function to α and a sealed version of the value 42, and unseals the result. This approach ensures that effectively-parametric code behaves as usual, but that the above example fails (because addition fails on sealed values):

$$\begin{aligned} & (\Lambda X. \lambda x X. x) [\mathbf{Int}] 42 \rightarrow^* \mathit{unseal}_\alpha(\mathit{seal}_\alpha(42)) \rightarrow^* 42 \\ & (\Lambda X. \lambda x X. ((x :: ?) + 1) :: X) [\mathbf{Int}] 42 \rightarrow^* \mathit{unseal}_\alpha(\boxed{(\mathit{seal}_\alpha(42) :: ?) + 1}) :: \alpha \rightarrow^* \mathit{error} \end{aligned}$$

Building upon work on dynamic enforcement of parametricity (Pierce and Sumii, 2000; Sumii and Pierce, 2004; Guha et al., 2007; Matthews and Ahmed, 2008), a first version of a parametric cast calculus¹ was proposed, albeit without any proof of parametricity (Ahmed et al., 2011). Years later, a variant of this cast calculus, λB (Ahmed et al., 2017), did come with a proof of parametricity. λB has also been used as a target language by Xie et al. (2018), who explore the treatment of *implicit* polymorphism. Another recent effort is System F_G , a gradual source language that is compiled to a cast calculus akin to λB , called System F_C (Igarashi et al., 2017a). These efforts highlight several key challenges and tensions in the design of a gradually parametric language (§2.4), and leave some questions unanswered, most notably the possibility to satisfy both parametricity and graduality.

The approaches mentioned before base their design on what we call *type-driven* sealing. Type-driven sealing decides whether to seal or not according to the type of the argument and the function being applied, generally preserving the familiar syntax of System F. Inspired by the challenges imposed by type-driven sealing, New et al. (2020) propose a *term-driven* sealing approach. Term-driven sealing is a different design that forgoes the familiar syntax of System F and instead relies on explicit terms for sealing and unsealing values. PolyG^v requires programmers to specify whether arguments should be sealed or not, and with which type variable, by writing for example $f [X = \text{Int}] (seal_X(42))$ and $f [X = \text{Int}] 42$, where f is a polymorphic function.² The first program explicitly seals argument 42 with the type variable X , while the second program does not. While this strategy has produced the first parametric gradual calculus, it is important to realize that this calculus does not solve the same problem as the one tackled by other proposals like λB (Ahmed et al., 2017) or System F_G (Igarashi et al., 2017a). Gradual languages are intended to smoothly support the static-to-dynamic checking spectrum, but as noted by New et al., PolyG^v supports this only when the untyped code already contains the right sealing annotations. In other words, in PolyG^v $f [X = \text{Int}] (seal_X(42))$ and $f [X = \text{Int}] 42$ are unrelated by precision, and therefore graduality does not relate their respective behavior. Although term-driving sealing allows PolyG^v to satisfy both parametricity and graduality, this comes at the cost of a different programming model with its own limitations (§2.4).

This thesis explores the integration of parametric polymorphism and gradual typing into a language while preserving parametricity, graduality, and the familiar syntax of System F. Chapter 2 starts with a detailed review of the challenges and tensions that affect the design of gradual parametric languages. As a result of the exploration, we develop GSF (Chapter 3), a type-driven gradual parametric language that bridges between System F and an untyped language with dynamic sealing primitives (Chapter 4). GSF is derived using AGT. The standard AGT application derives a polymorphic gradual language that satisfies graduality

¹ The difference between a gradual (source) language and a cast calculus is that a cast calculus demands explicit use of casts in order to exploit the flexibility of runtime checking—whether a cast calculus is meant to be used directly by programmers is in the eyes of the beholder. A gradual source language is usually defined by a type-directed translation to a cast calculus, with its own typing and reduction rules. In the AGT methodology reduction is instead defined directly for the gradual source language, in terms of the reduction of enriched typing derivations (Garcia et al., 2016). Toro and Tanter (2017, 2020) prove that both approaches are equivalent in a standard, simply-typed setting.

² As the syntax suggests, type variables in PolyG^v are introduced at instantiation time, with outward scoping; this requires linear typing environments and a mechanism to limit their propagation to the current lambda abstraction (New et al., 2020).

but breaks parametricity. Therefore, some adjustments in the runtime semantics must be made to recover this property. Unfortunately, graduality is violated in the new language because the implemented changes to recover parametricity are in conflict. Nevertheless, we state a weaker property that allows us to characterize the static-to-dynamic checking spectrum afforded by GSF. Despite its lack of full support for graduality, GSF goes further than previous work based on the type-driven sealing approach since it has simple static semantics and satisfies the embedding-projection pairs along with other expected properties studied in detail in Chapter 2. We also present a detailed study on the different forms of gradual parametricity, shedding light on the current design space. Chapter 5 extends GSF with existential types, which are the core of data abstraction mechanisms. The extension of GSF with existential types is also derived using AGT, and similar adjustments to those implemented by GSF have to be done to recover parametricity, thus satisfying parametricity and a weak dynamic gradual guarantee.

Following the study and the implementation of GSF, and inspired by the pending challenges of the type-driven and term-driven approaches to facing gradual parametricity, we propose plausible sealing, developing the language Funky (Chapter 6). Funky is a term-driven sealing language that, in addition to allowing specifying whether an argument is sealed or not regarding a specific type variable (like PolyG^ν), it enables specifying maybe-sealed values, *i.e.* the function argument can be used as a sealed value at any type variable in scope, but not necessarily. It is at runtime, depending on the context in which they are used, that it is determined whether to use these maybe-sealed values sealed or not. Furthermore, Funky avoids global dynamic sealing and, instead, proposes a novel lexically-scoped form of sealing realized using a representation of evidence inspired by the category of spans. Consequently, Funky satisfies a standard formulation of parametricity that does not break System F equivalences but loses the property of embedding an untyped language with dynamic sealing primitives, as is the case of GSF. Plausible sealing makes it possible to use Funky as the elaboration target of different gradual source languages; we describe one such source language, Funk, with the familiar syntax of System F. We formally develop our approach in a simplified setting where polymorphism is restricted to instantiations with base and variable types. We prove that Funky satisfies both parametricity and graduality.

This thesis contributes to adopting the widely used type abstraction mechanism, parametric polymorphism, in a gradual language. We explore the integration of these two features in-depth by deriving two languages, GSF and Funky. We look at what properties we obtain and the limitations of each proposal, thus developing the metatheory of gradual parametricity. In addition, we study the different meanings that parametricity can have in a gradual language. While the results presented in this thesis still have some constraints, they propose a novel way to reconcile parametricity and graduality in a language with the System F syntax that can be potentially reusable in other settings.

Publications. Chapter 3, “Gradual System F”, corresponds to a published work in the Proceedings of the ACM on Programming Languages 3, POPL (Jan. 2019), 17:1–17:30 (Toro et al., 2019), in which the author of this thesis served as the second author. Chapter 3 extends this publication with a detailed analysis of the dynamic gradual guarantee violation and the development of the weaker gradual guarantee that GSF satisfies (§3.6). Also, the presentation of gradual parametricity (3.7) includes a new comparison between the ap-

proaches of Ahmed et al. (2017) and New et al. (2020), shedding light on the current design space. Together with these extended contents, Chapters 4 (“Embedding Dynamic Sealing in GSF”) and 5 (“Gradual Existential Types in GSF”), were published (“Gradual System F”) in the Journal of the ACM (JACM). In this work, the author of this thesis acted as the first author. Finally, Chapter 6 and 7 (“Plausible Sealing for Gradual Parametricity”) was published in the Proceedings of the ACM on Programming Languages 6, OOPSLA1 (April 2022), 70:1–70:28 (Labrada et al., 2022), in which the author of this thesis acted as the first author.

Chapter 2

Gradual Parametricity: Background and Basics

Type systems are a powerful mechanism to help ensure certain properties and guarantees in programming languages. For example, the simply typed lambda-calculus satisfies type safety (the evaluation of a well-typed program does not get stuck) and normalization (the evaluation of a well-typed program is guaranteed to halt in a finite number of steps). Likewise, a language with parametric polymorphism, such as System F satisfies both of the above properties and also a rich semantics property known as relational parametricity (a polymorphic value must behave uniformly for all possible instantiations). In this thesis, we focus on polymorphic types in a gradual type system so that we start with an introduction to parametric polymorphism and parametricity, as well as gradual typing. We then briefly motivate gradual parametricity through a basic example. We conclude this chapter by identifying several design issues in existing gradual languages.

2.1 Parametric Polymorphism

Parametric polymorphism enables the definition of terms that can operate over any type, with the introduction of type variables and universally-quantified types. For instance, a function of type $\forall X.X \rightarrow X$ can be used at any type, and returns a value of the same type as its actual argument. For the sake of this work, it is important to recall two crucial distinctions that apply to languages with parametric polymorphism, one syntactic—whether polymorphism is explicit or implicit—and one semantic—whether polymorphic types impose strong behavioral guarantees or not.

Explicit vs Implicit. In a language with *explicit* polymorphism, such as the Girard-Reynolds polymorphic lambda calculus (*a.k.a.* System F) (Girard, 1972; Reynolds, 1974), the term language includes explicit type abstraction $\Lambda X.e$ and explicit type application $e [T]$, as illustrated next:

```
let f :  $\forall X.X \rightarrow X$  =  $\Lambda X.\lambda x:X.x$  in f [Int] 10
```


The function \mathfrak{f} has the polymorphic (or universal) type $\forall X.X \rightarrow X$. By applying \mathfrak{f} to type Int (we also say that \mathfrak{f} is *instantiated* to Int), the resulting function has type $\text{Int} \rightarrow \text{Int}$; it is then passed the number 10. Hence the program evaluates to 10.

In contrast to this explicit, Church-style formulation, the Curry-style presentation of *polymorphic type assignment* (Curry et al., 1972) does not require type abstraction and type application to be reflected in terms. This approach, known as *implicit* polymorphism, has inspired many languages such as ML and Haskell, which use the traditional Damas-Milner type system (Damas and Milner, 1982). This type system can generalize the type of a term and can give a term of polymorphic type a (partially) instantiated type. Alternatively, implicit polymorphism can be seen as inducing a notion of subtyping relating polymorphic types to their instantiations (Mitchell, 1988; Odersky and Läufer, 1996); *e.g.* $\forall X.X \rightarrow X <: \text{Int} \rightarrow \text{Int}$. Implicitly-polymorphic languages are often compiled into an explicitly-polymorphic language. For instance, the use of the subtyping judgment $\forall X.X \rightarrow X <: \text{Int} \rightarrow \text{Int}$ is materialized by introducing an explicit instantiation $[\text{Int}]$, and vice-versa, the use of the judgment $\text{Int} \rightarrow \text{Int} <: \forall X.\text{Int} \rightarrow \text{Int}$ is materialized by inserting a type abstraction constructor ΛX .

Genericity vs. Parametricity. Some languages with universal type quantification also support intensional type analysis or reflection, which allows a function to behave differently depending on the type to which it is instantiated. For instance, in Java, a generic method of type $\forall X.X \rightarrow X$ can use `instanceof` to discriminate the actual type of the argument, and behave differently for `String`, say, than for `Integer`. Therefore these languages only support *genericity*, *i.e.* the fact that a value of a universal type can be safely instantiated at any type.¹

Parametricity is a much stronger interpretation of universal types, which dictates that a polymorphic value *must behave uniformly* for all possible instantiations (Reynolds, 1983). This implies that one can derive interesting theorems about the behavior of a program by just looking at its type, hence the name “free theorems” coined by Wadler (1989). For instance, one can prove using parametricity that any polymorphic function of type $\forall X.\text{List } X \rightarrow \text{List } X$ commutes with the polymorphic map function. Technically, parametricity is expressed in terms of a (type-indexed) *logical relation* that denotes when two terms behave similarly when viewed at a given type. All well-typed terms of System F are related to themselves in this logical relation, meaning in particular that all polymorphic terms behave uniformly at all instantiations (Reynolds, 1983).

Simply put, if a value \mathfrak{f} has type $\forall X.X \rightarrow X$, then—modulo divergence if admitted in the considered language—genericity only tells us that $\mathfrak{f} \ [\text{Int}] \ 10$ reduces to *some* integer, while parametricity tells the much stronger result that $\mathfrak{f} \ [\text{Int}] \ 10$ necessarily evaluates to 10, *i.e.* \mathfrak{f} has to be the identity function. In the context of gradual typing, Ina and Igarashi (2011) have explored genericity with a gradual variant of Java. All other work has focused on the challenge of enforcing parametricity (Ahmed et al., 2011, 2017; Igarashi et al., 2017a; Xie et al., 2018; New et al., 2020).

¹ We call this property *genericity*, by analogy to the name *generics* in use in object-oriented languages like Java and C#.

2.2 Gradual Typing

Static and dynamic typechecking have dual advantages and limitations. For instance, adopting a static discipline provides early detection of errors at the expense of conservatively rejecting some programs that would go right. On the other hand, adopting a dynamic discipline provides flexibility at the cost of extra checks (and errors!) at runtime. Gradual typing is a specific approach to combine static and dynamic checking within the same language, letting programmers control which checking discipline is used where, and supporting the convenient evolution between both (Siek and Taha, 2006). Specifically, programmers can use the unknown type $?$ to denote the absence of statically-known type information. Hence, a program without any $?$ is a statically-typed program, and a program where all binders and constants have type $?$ is a dynamically-typed program. In between, there is a whole spectrum of flexibility according to the programmers needs.

Precision and consistency. To support the transition between static and dynamic typechecking, gradual languages rely on an important relation between types called *(im)precision*, which intuitively denotes how much is known about a given type. Of course $?$ is the least precise type. In the gradually-typed lambda calculus GTLC, type precision is defined as Siek and Taha (2006); Siek et al. (2015a):

$$\frac{}{B \sqsubseteq B} \qquad \frac{G_1 \sqsubseteq G_2 \quad G'_1 \sqsubseteq G'_2}{G_1 \rightarrow G'_1 \sqsubseteq G_2 \rightarrow G'_2} \qquad \frac{}{G \sqsubseteq ?}$$

Where B stands for base types such as `Int` and `Bool`, and G stands for gradual types. For instance, `Int` \rightarrow `Bool` \sqsubseteq `?` \rightarrow `Bool` \sqsubseteq `?` \rightarrow `?` \sqsubseteq `?`. Observe that, unlike subtyping, precision is covariant for both the domain and codomain of function types. Precision on terms, noted $t_1 \sqsubseteq t_2$, is the natural lifting of type precision to terms:

$$\frac{}{b \sqsubseteq b} \qquad \frac{}{x \sqsubseteq x} \qquad \frac{G_1 \sqsubseteq G_2 \quad t_1 \sqsubseteq t_2}{\lambda x : G_1.t_1 \sqsubseteq \lambda x : G_2.t_2} \qquad \frac{t_1 \sqsubseteq t_2 \quad t'_1 \sqsubseteq t'_2}{t_1 t'_1 \sqsubseteq t_2 t'_2}$$

For instance, `$\lambda x : \text{Int}.x$` \sqsubseteq `$\lambda x : ? .x$` .

A gradual type system optimistically deals with imprecision, thereby relaxing standard relations on types. For instance, type equality is relaxed as type *consistency* \sim . Two types are consistent if they agree on their known parts. For instance `Int` \rightarrow `?` \sim `?` \rightarrow `Bool`, but `Int` $\not\sim$ `Bool`. Type consistency can be formally defined as Siek and Taha (2006):

$$\frac{}{B \sim B} \qquad \frac{G_1 \sim G_2 \quad G'_1 \sim G'_2}{G_1 \rightarrow G'_1 \sim G_2 \rightarrow G'_2} \qquad \frac{}{? \sim G} \qquad \frac{}{G \sim ?}$$

Regarding the dynamics semantics, the standard approach consists in elaborating gradual source terms via a typed-driven translation to a *cast calculus*, a core language with explicit

runtime typechecks. The translation inserts casts at the boundaries between static and dynamic typing, ensuring at runtime that violations of static assumptions are detected and manifest as errors. For instance, the well-typed source gradual term $(\lambda x : ?.x + 1) \text{ true}$ ($?$ is consistent with `Bool`) would typically be elaborated to the target term $(\lambda x : ?.\langle \text{Int} \Leftarrow ? \rangle x + 1) (\langle ? \Leftarrow \text{Bool} \rangle \text{true})$. In this target program, the cast $\langle \text{Int} \Leftarrow ? \rangle$ requires that the argument given at runtime is indeed an `Int` value, and $\langle ? \Leftarrow \text{Bool} \rangle$ is used to match the parameter and the function argument types. This program reduces to $(\langle \text{Int} \Leftarrow ? \rangle (\langle ? \Leftarrow \text{Bool} \rangle \text{true}) + 1)$, failing in the next reduction step when it tries to combine the cast $\langle ? \Leftarrow \text{Bool} \rangle$ with $\langle \text{Int} \Leftarrow ? \rangle$ (`Bool` is not consistent with `Int`).

Illustration. Let us consider the following three programs A, B and C.

A) $\text{let } x : ? = \text{true in } (\lambda y : ?.y + 1) x$ B) $\text{let } x : ? = \text{true in } (\lambda y : \underline{\text{Int}}.y + 1) x$ C) $\text{let } x : \underline{\text{Bool}} = \text{true in } (\lambda y : \underline{\text{Int}}.y + 1) x$

All three programs first bind `true` to `x` and then pass `x` as argument to a function that adds one to its argument. They only differ in their precision: *i.e.* the type annotations range from all statically unknown (A) to all known (C). Program A typechecks and fails at runtime when trying to add 1 to `true`. Program B is a more precise variant in which the function argument type is now declared to be `Int`. This program also fails at runtime, but it does so *earlier* than program A: the error is detected when trying to apply the function to `true`. Finally, program C is fully static, and is ill-typed. So, by augmenting the precision of a program, we may go from failing at runtime to failing statically.

Properties of gradual languages. To characterize the static-to-dynamic checking spectrum afforded by gradual typing, Siek et al. (2015a) summarized and extended the expected properties of gradual languages, recalled hereafter:

- *Type safety* establishes that well-typed programs cannot get stuck, although they can produce runtime errors due to actual violations of (optimistic) assumptions made during type checking.
- *Conservative extension of a static discipline* means that fully-precise terms typecheck and evaluate exactly as they would in the static language. Of course, this criterion is relative to which language is considered as the “static end” of the spectrum.
- *Embedding of a dynamic discipline* characterizes the capability of the gradual language to accommodate (possibly through a syntactic translation) terms of a dynamically-checked language. Like conservative extension, this criterion is relative, this time with respect to the “dynamic end” of the spectrum.
- *Gradual guarantees.* The gradual guarantees capture the smoothness of the static-to-dynamic checking spectrum, requiring both typing (SGG) and evaluation (DGG) to be monotonic with respect to imprecision. Specifically, if a program is well typed, then a less precise version should also be well typed; likewise, if a program runs to completion without errors, so should a less precise version.

In addition to these key formal properties, there are other interesting aspects not explicitly addressed by Siek et al. (2015a) that are worth considering.

- *Harmless imprecise ascriptions.* New and Ahmed (2018) give a semantic account of the dynamic gradual guarantee, called *graduality*, based on the notion that imprecision induces embedding-projection pairs. A particular consequence of their formulation is that imprecise ascriptions are harmless: given a term $t : A$ and $A \sqsubseteq B$, then $t :: B :: A$ is equivalent to t . Observe that this property is weaker than the DGG, as the latter is not restricted to outer ascriptions. Note that the harmless imprecision property is an implication of the DGG if we assume that t is equivalent to $t :: A$ and, therefore, to $t :: A :: A$. Assuming that t is equivalent to $t :: A :: A$ and that $t :: A :: A$ is more precise than $t :: B :: A$, by the DGG, we get that $t :: A :: A$ is equivalent to $t :: B :: A$. Therefore, by transitivity, t is equivalent to $t :: B :: A$.
- *Expressiveness of imprecision.* A gradual language soundly augments the expressiveness of the original static type system. Let us illustrate what we mean in a simply-typed setting (STLC refers to the simply-typed lambda calculus with base types), and how imprecision allows bridging the gap towards System F:
 1. Consider the STLC term $t = \lambda x : T.x$, *i.e.* the identity function for values of some type T . The term t is operationally valid at different types, but it cannot be given a general type in STLC. Its type has to be fixed at either $\text{Int} \rightarrow \text{Int}$, $\text{Bool} \rightarrow \text{Bool}$, etc.
 2. Intuitively, a proper characterization of t requires going from simple types to parametric polymorphism, such as System F. In System F, we could use the type $\forall X.X \rightarrow X$ to precisely specify that t can be applied with any argument type and return the same type.
 3. With a gradual variant of STLC, we can give term t the imprecise type $? \rightarrow ?$ to statically capture the fact that t is definitely a function, without committing to specific domain and codomain types.
 4. This lack of precision is soundly backed by runtime enforcement, such that the term $(t\ 3)\ 1$ evaluates to a runtime type error.

Abstracting Gradual Typing. The gradualization of a language presents some challenging steps. One of them is defining the notion of consistency between types, which becomes more difficult as type systems become more complex. In addition, it is generally used a casts calculus to mediate between statically and dynamically typed code, which is designed mostly by intuition (Garcia et al., 2016). In order to assist language designers in crafting new gradual languages, Garcia et al. (2016) proposed the Abstracting Gradual Typing methodology (AGT, for short). The promise of AGT is that, starting from a specification of the *meaning* of gradual types in terms of the set of possible static types they represent, one can systematically derive all relevant notions, including precision, consistent predicates (*e.g.* consistency and consistent subtyping), consistent functions (*e.g.* consistent meet and join), as well as a direct runtime semantics for gradual programs, obtained by reduction of gradual typing derivations augmented with evidence for consistent judgments. An essential advantage of this methodology is that a gradual language built with it achieves, by construction, the refined criteria for gradual typing (Siek et al., 2015a).

The AGT methodology has so far proven effective to assist in the gradualization of a number of disciplines, including effects (Bañados Schwerter et al., 2014, 2016), record subtyping (Garcia et al., 2016), set-theoretic types (Castagna and Lanvin, 2017), union types (Toro and Tanter, 2017), refinement types (Lehmann and Tanter, 2017) and security types (Toro et al., 2018). The applicability of AGT to gradual parametricity is an open question repeatedly raised in the literature—see for instance the discussions of AGT by Igarashi et al. (2017a) and Xie et al. (2018). Considering the variety of successful applications of AGT, and the complexity of designing a gradual parametric language, in this work we decide to adopt this methodology, and report on its effectiveness.

2.3 Gradual Parametricity in a Nutshell

Gradual parametricity ought to support *imprecise* type information while ensuring that assumptions about *parametricity* are enforced at runtime whenever they are not definitely provable statically. To illustrate, consider the following program:

```
let g: ? = [λa.λb.if b then a else a + 1] in
let f: ∀X.X→X = λX.λx:X.g x [v] in
f [Int] 10
```

Function `f` is given the polymorphic type $\forall X.X \rightarrow X$, and is therefore expected to behave parametrically. It is then instantiated at type `Int`, and applied to the value `10`. Note that `f` is implemented using a function `g` of unknown type, which is the result of embedding (`[·]`) untyped code into the gradual language. We write `[v]` to stand for the value `true` or `false`.

While this program is gradually well-typed, the compliance of `f` with respect to its declared parametric behavior is unknown statically. By parametricity, `f` should behave as the identity function (§2.1). But `g` itself behaves as an identity function only if its second argument `[v]` is `true`. Conversely, if `[v]` is `false`, a runtime error is raised to report the parametricity violation.

This example highlights two key characteristics of gradual parametricity. First, to enforce parametricity gradually requires more than tracking type safety. If we let the program reduce to `11` when `[v]` is `false`, then type *safety* is not endangered; only type *soundness* (*i.e.* parametricity) is. Second, the statement of free theorems must account for the possible effects of the gradual language: gradual programs can produce runtime errors—and usually can also diverge even if the corresponding static language is strongly normalizing (Siek and Taha, 2006).

Also, gradualizing a language with parametric polymorphism requires extending the notion of precision to account for both type variables and polymorphic types. The natural definition of precision simply proceeds congruently:

$$\frac{}{X \sqsubseteq X} \qquad \frac{G_1 \sqsubseteq G_2}{\forall X.G_1 \sqsubseteq \forall X.G_2}$$

and likewise for terms:

$$\frac{t \sqsubseteq t'}{\Lambda X.t \sqsubseteq \Lambda X.t'} \qquad \frac{t \sqsubseteq t' \quad G \sqsubseteq G'}{t [G] \sqsubseteq t' [G']}$$

As we will see, this natural extension of precision to System F is not the only one that has been (and will be) considered, as it exposes a deep tension between parametricity and the gradual guarantees.

2.4 Gradual Parametricity: Challenges

While the basics of gradual parametricity illustrated in Section 2.3 are uncontroversial, the devil is in the details. There are fundamental tensions in the design of gradual parametricity that arise from the desirable metatheoretical properties—both of parametricity and of gradual typing—which do not seem to be simultaneously satisfiable. We now explain how existing languages in the gradual parametricity design space differ, covering each desirable property described in the previous sections, plus a few properties previously not covered.

Parametricity. Establishing that a gradual parametric language enforces parametricity has been a long-standing open issue: early work on the polymorphic blame calculus did not prove parametricity (Matthews and Ahmed, 2008; Ahmed et al., 2009b, 2011), and the first parametricity result was established several years later for a variant of that calculus, λB (Ahmed et al., 2017). In fact, λB is a cast calculus, not a gradual source language, meaning that explicit casts should be sprinkled in the program above to achieve the same result. Igarashi et al. (2017a) develop a gradual source language, System F_G , whose semantics are given by translation to a cast calculus, System F_C , which is a close cousin of λB . Igarashi *et al.* do not prove parametricity, but conjecture that due to the similarity between System F_C and λB , parametricity should hold. Xie et al. (2018) develop a language (here referred to as CSA) with implicit polymorphism, which compiles to λB and therefore inherits its parametricity result. More recently, New et al. (2020) explore a radically different point in the design space with PolyG ^{ν} , a language that requires explicit *sealing* and *unsealing* terms. This choice sidesteps some tensions and yields a notion of gradual parametricity that is stronger and more faithful to the original presentation of Reynolds than that of prior work. We come back to the discussion of different notions of gradual parametricity, which gets fairly technical, in §3.7.

Conservative Extension of System F. Most work on gradual parametricity— λB ,² System F_G , CSA, as well as the present work—consider System F as the starting point, meaning that System F programs should be valid programs in these languages, and behave as they would in System F. System F_G is a conservative extension of System F, and CSA of an implicitly-polymorphic variant of System F (Damas and Milner, 1982). λB is a cast calculus whose syntax of fully-precise cast-free terms also coincides with System F. For such terms, λB is also a conservative extension of System F. However, its compatibility relation for types is not a conservative extension of type equality in System F. For instance, the polymorphic type $\forall X.X \rightarrow X$ is not only compatible with $\text{Int} \rightarrow \text{Int}$ —a defining feature of *implicit* polymorphism—it is also compatible with $\text{Int} \rightarrow \text{Bool}$. PolyG ^{ν} departs from the syntax of

² Although λB is a polymorphic cast calculus, we include it in the discussion of gradual languages in this work since all other gradual languages like System F_G and CSA translate to λB to establish their dynamic semantics and properties.

System F and so is not a conservative extension of it. For instance, the following System F program defines a function f , which is the identity function, and instantiates it at type Int , applies it to 1, and then adds 1 to the result, yielding 2.

```
let f :  $\forall X.X \rightarrow X = \lambda X.\lambda x:X.x$  in (f [Int] 1) + 1
```

This program is rejected statically in PolyG^ν , because the sealing and unsealing that implicitly underlies polymorphic behavior in System F must happen *explicitly* in the syntax of terms, resulting in the more verbose program:

```
let f :  $\forall X.X \rightarrow X = \lambda X.\lambda x:X.x$  in unseal $_X$ (f [X=Int] (seal $_X$  1)) + 1
```

PolyG^ν forces an outward scoping of type variables, *i.e.* $[X=\text{Int}]$ above puts x in scope for subsequent use by seal_x and unseal_x . While not addressed by the authors, it seems reasonable to conjecture that PolyG^ν is a conservative extension of such an unusual static source language with explicit sealing.

Embedding of a Dynamic Language. Because a polymorphic language includes a simply-typed core, one naturally expects an untyped lambda calculus to be embeddable in a gradual polymorphic language (Ahmed et al., 2011). As we will see in Chapter 4, and previously explored by Siek and Wadler (2016), the dynamic end of the spectrum for a gradual parametric language can be even more interesting, accommodating dynamic sealing primitives (Sumii and Pierce, 2004).

Gradual Guarantees. A major tension faced by gradual parametric languages in the past decade has been to attempt to reconcile the gradual guarantees with parametricity. While this thesis will dive into this question repeatedly and at a quite technical level, let us present here what happens on the surface, for a programmer. Consider this program, which is the same as above, except that the return type of the function f is now unknown:

```
let f :  $\forall X.X \rightarrow ? = \lambda X.\lambda x:X.x$  in (f [Int] 1) + 1
```

Following the motto that imprecision is harmless, a programmer might expect this program to both typecheck and run without errors, yielding 2. However, in λB and System F_G , the (elaboration with casts of the) above program fails with a runtime error, because the result of f [Int] 1 is *sealed*, and therefore unusable directly.

This failure is a violation of the dynamic gradual guarantee (DGG). But in fact, technically, this behavior only is a violation if we consider the program above to actually be a more imprecise variant of the System F program where the return type of f is x instead of $?$. Faced with this tension between the gradual guarantees and parametricity, Igarashi et al. (2017a) introduce a *stricter* notion of precision in System F_G , which does not allow losses of precision in *parametric* positions of a polymorphic type. For instance, in System F_G , $\forall X.X \rightarrow \text{Int}$ is considered more precise than $\forall X.X \rightarrow ?$, but $\forall X.X \rightarrow X$ is not. Igarashi et al. (2017a) prove the static gradual guarantee for System F_G based on this more restrictive notion of precision, but leave the corresponding dynamic gradual guarantee as a conjecture.

PolyG^ν addresses this tension between parametricity and the dynamic gradual guarantee by uncoupling sealing and precision, using a syntax with explicit sealing and unsealing. If we start with the following fully-static program:

```
let f :  $\forall X.X \rightarrow X$  =  $\Lambda X.\lambda x:X.x$  in unsealx(f [X=Int] (sealx 1)) + 1
```

Then making the return type of `f` unknown yields a program that still typechecks and runs successfully. This is clear because the sealing behavior that was causing problem is now explicit in the terms, and therefore not affected by a loss of precision in types. However, this choice of syntax is not innocuous. Consider the following imprecise program, where the body of `f` has been elided:

```
let f :  $\forall X.X \rightarrow ?$  = body in unsealx(f [X=Int] (sealx 1)) + 1
```

If body is $\Lambda X.\lambda x:X.x$, then the program evaluates to 2 as expected. However, if `f` is a constant function, *e.g.* body is $\Lambda X.\lambda x:X.1$, then this PolyG^ν program fails because the call-site unsealing of the value returned by `f` is now invalid. If one removes `unsealx` around the application of `f`, the program evaluates to 2, *i.e.* `f` behaves as a constant function. But now, the case where body is the polymorphic identity function fails, when trying to add 1 to a sealed value. This means that in PolyG^ν, the decision to use unsealing or not at a call site cannot be taken *modularly*: one needs to know the implementation of `f` to decide.

Faithful Type Instantiations. What should type instantiations on terms of unknown type mean? Below, the polymorphic identity function ends up instantiated to `Int` and passed a `Bool` value:

```
let g : ? =  $\Lambda X.\lambda x:X.x$  in g [Int] true
```

This program in System F_G, and a possible adaptation to λB (following the translation proposed by Igarashi et al. (2017a)), both return `true` despite the explicit instantiation to `Int`. Internally, this happens because `g` is first consistently considered to be of type $\forall X.?$ in order to accommodate the type instantiation, but then the instantiation yields a substitution of `Int` for `X` in `?`, which in both languages is just `?`. There is no tracking of the decision to instantiate the underlying value to `Int`.

In contrast, if we try to write a similar program in PolyG^ν:

```
let g : ? =  $\Lambda X.\lambda x:X.x$  in g [X=Int] (sealx true)
```

Then this program does not even typecheck, because sealing `true` with `x` requires the types to coincide. If we ascribe `true` to the unknown type before sealing it, then the program typechecks but fails at runtime, thereby respecting the type instantiation to `Int`.

Expressiveness of Imprecision. We can unfold the exact same line of reasoning presented in §2.2, but this time starting from System F and bridging the gap towards System F_ω:

1. Consider the System F term $t = \lambda x : \forall X.T.(x \text{ [Int]})$, which behaves as an instantiation function to `Int`, for some polymorphic type $\forall X.T$. The term t is operationally valid at different types, but cannot be given a general type in System F. Its type has to be fixed at either $(\forall X.X \rightarrow X) \rightarrow (\text{Int} \rightarrow \text{Int})$ (where $\forall X.T = \forall X.X \rightarrow X$), $(\forall XY.X \rightarrow Y \rightarrow X) \rightarrow (\forall Y.\text{Int} \rightarrow Y \rightarrow \text{Int})$ (where $\forall X.T = \forall XY.X \rightarrow Y \rightarrow X$), etc.

2. Intuitively, a proper characterization of t requires going from System F to higher-order polymorphism, such as System F_ω . In System F_ω , we could use the type $\forall P.(\forall X.P X) \rightarrow (P \text{Int})$ to precisely specify that t instantiates any polymorphic argument to Int . Term t can be expressed in System F_ω as $t_\omega = \Lambda P.\lambda x : (\forall X.P X).x [\text{Int}]$.
3. With a gradual variant of System F, we ought to be able to give term t the imprecise type $(\forall X.?) \rightarrow ?$ to statically capture the fact that t is definitely a function that operates on a polymorphic argument, without committing to a specific domain scheme and codomain type.
4. This lack of precision ought to be soundly backed by runtime enforcement, such that, given $id : \forall X.X \rightarrow X$, the term $(t id) \text{true}$ should evaluate to a runtime type error. (Observe that in System F_ω , $(t_\omega [\forall X.X \rightarrow X] id) \text{true}$ is ill-typed.)

The fact that λB and System F_C do not respect type instantiations on imprecise types mean that in these systems, the term $(t id) \text{true}$ does not raise any error.³ Therefore, while these higher-order polymorphic patterns can be expressed, they are unsound.

Now consider the same example adapted to PolyG^ν :

```
let t : ( $\forall X.?$ )  $\rightarrow$  ? =  $\lambda x : (\forall X.?). x [x=\text{Int}]$  in (t id) (sealx true)
```

This program does not typecheck in PolyG^ν because the type variable x used explicitly in the body of the function is *no longer in scope* at the use site to seal the value true . Recall that $[x=\text{Int}]$ puts x in scope for the rest of the *lexical* scope of the instantiation, but it does not cross function boundaries. So, in addition to the modularity issues presented in the previous section, the explicit (un)sealing mechanism of PolyG^ν cannot accommodate higher-order patterns like the above, which requires abstracting over type applications.

Polymorphic Interoperability. λB , System F_G , and PolyG^ν are languages with *explicit* polymorphism, *i.e.* with explicit type abstraction and type application terms. Despite this, λB and System F_G accommodate some form of implicit polymorphism, with different flavors. The underlying motivation is to support *interoperability* between typed and untyped code, considering that type abstraction and application are meaningless terms in an untyped language. The archetypal example is the System F polymorphic identity function, which one would like to be able to use in untyped code as standard function, or vice versa, using the untyped identity function as a polymorphic one.

λB features two type compatibility rules to support this kind of implicit polymorphism:

$$\text{(Comp-AllR)} \frac{\Sigma; \Delta, X \vdash T_1 <: T_2 \quad X \notin T_1}{\Sigma; \Delta \vdash T_1 <: \forall X.T_2} \quad \text{(Comp-AllL)} \frac{\Sigma; \Delta \vdash T_1[?/X] <: T_2}{\Sigma; \Delta \vdash \forall X.T_1 <: T_2}$$

These rules permit $\forall X.X \rightarrow X$ to be compatible with $? \rightarrow ?$, but as first identified by Xie et al. (2018) and recalled above, these rules also imply that the type $\forall X.X \rightarrow X$ is compatible with both $\forall X.\text{Int} \rightarrow \text{Bool}$ and $\text{Int} \rightarrow \text{Bool}$. System F_G does not relate $\forall X.X \rightarrow X$ with any

³In System F_C , $(t id) \text{true}$ fails because $\forall X.?$ is not deemed consistent with $\forall X.X \rightarrow X$. Consequently, t must be declared to take an argument of type $?$ instead of $\forall X.?$. The result is the same as in λB however: no runtime error is raised.

	Polym	SF	TS	Param	CE	ED	SGG	DGG	HIA	FTI	EI	PI
λB	mixed	✓	✓	✓	✓	✓ σ	-	✗	✗	✗	✗	✓
System F_G	mixed	✓	✓	c	✓	✓	w	c (w)	-	✗	✗	✓
CSA	implicit	✗	✓	c	✓ s	-	✓	✗	-	-	-	✓
Poly G^ν	explicit	✗	✓	✓	na	-	-	✓	✓	✓	✗	✗
GSF	explicit	✓	✓	✓	✓	✓ σ	✓	w	✓	✓	✓	✗
$F_\varepsilon^?$	explicit	✗	✓	✓	na	✓	✓	✓	✓	✓	✓	✗

Table 2.1: Comparison of approaches to gradual parametricity.

Polym: form of polymorphism. SF: System F syntax. TS: type safety. Param: parametricity. CE: conservative extension. ED: embedding of dynamic language. SGG: static gradual guarantee. DGG: dynamic gradual guarantee. HIA: harmless imprecise ascriptions. FTI: faithful type instantiations. EI: expressiveness of imprecision. PI: polymorphic interoperability. ✓: the property has been proven. ✗: the property is not satisfied. -: the property is not studied. na: the property does not apply. ✓ s : proved only for the static semantics. ✓ σ : can embed untyped lambda calculus with dynamic sealing. c: the property is explicitly conjectured but not proven. w: guarantees stated wrt a restricted precision.

of its static instantiations. However, it *does* relate that type with $? \rightarrow ?$, considered to be *quasi-polymorphic*, on the basis that using the unknown type should bring some of the flexibility of implicit polymorphism.

Xie et al. (2018) argue that it is preferable to clearly separate the subtyping relation induced by implicit polymorphism from the consistency relation induced by gradual types. As a result, CSA features intuitive and straightforward definitions of precision and consistency, while accommodating the flexibility of implicit polymorphism in full.

Summary. Table 2.1 summarizes the different approaches we reviewed. The last two lines correspond to our proposals, GSF and $F_\varepsilon^?$, which are formally developed in the following chapters. A check denotes a property that is proven. A cross denotes a property that is not satisfied. A question mark is used for results that are explicitly conjectured, while a dash is used for results that are not studied. For the conservative extension result (CE) for CSA, the “s” signals that the result is only established with respect to typing, not reduction. For the embedding of a dynamic language (ED), we annotate with a “ σ ” when the language has been shown to embed an untyped lambda calculus with sealing primitives (Siek and Wadler (2016) for λB , and §4.4 for GSF). We use “w” (weak) to denote gradual guarantees stated with respect to a stricter notion of precision than the natural one.

Note that languages CSA, Poly G^ν , and $F_\varepsilon^?$ are not based on System F syntax (SF). CSA is a language with only implicit polymorphisms, so it does not use the syntax of System F. Poly G^ν is a language with sealing/unsealing primitives, and $F_\varepsilon^?$ is an evidence-based language. The use of these features in both languages is necessary to obtain programs equivalent to the ones in System F. All languages prove the type safety (TS) property. The parametricity (Param) property is left as a conjecture for System F_G and CSA. The conservative extension (CE) does not apply for Poly G^ν and $F_\varepsilon^?$ since they are not based on the System F syntax. Related to the embedding of a dynamic language (ED), System F_G and $F_\varepsilon^?$ embed the untyped lambda calculus, and λB and $F_\varepsilon^?$ an untyped lambda calculus with sealing primitives. For the other languages, the property is not studied. The static gradual guarantee (SGG) is not

studied for λB and PolyG^ν , is weaker in System F_G since it is based on a restricted notion of type precision and is satisfied for the other proposals. The dynamic gradual guarantee is only satisfied by PolyG^ν and $F_\varepsilon^?$ based on the natural notion of precision, is proved for GSF and left as a conjecture in System F_G but based on a restricted precision. Harmless imprecise ascription (HIA) and faithful type instantiation (FTI) are satisfied for PolyG^ν , GSF, and $F_\varepsilon^?$. Expensiveness of imprecision (EI) is satisfied by GSF and $F^?$. It is worth pointing out that EI is not a formal property; it is studied based on the example given earlier in this section. PolyG^ν , GSF, and $F_\varepsilon^?$ do not support interoperability between typed and untyped code.

Given the several design issues in existing gradual languages identified in this section, the next chapter proposes GSF, a gradual parametric language in the style of System F by applying AGT.

Part I

Gradual Parametricity, Revisited

Chapter 3

Gradual System F

This chapter explores the integration of parametric polymorphism and gradual typing into a language following the type-driven sealing approach. It proposes a gradual parametric language in the style of System F by applying the Abstracting Gradual Typing methodology (AGT) (Garcia et al., 2016). The resulting language, called GSF (for Gradual System F), embodies a number of important design choices. The first is in its name: it is an extension of System F, and therefore sticks to the traditional syntax of the polymorphic lambda calculus, where terms need not bother with sealing explicitly as in PolyG^ν . Two major characteristics differentiate GSF from other gradual parametric languages based on System F. First, GSF ensures that type instantiations on imprecise types are faithfully supported, thereby soundly supporting higher-order polymorphic programming patterns. Second, System F polymorphic values in GSF can flow into imprecise code while preserving their original behavior, because imprecise ascriptions are harmless.

We present a quick tour of GSF, including its design principles and main properties (§3.1). We then explain how we derive GSF from a variant of System F called SF (§3.2), by following AGT. While mostly standard, SF is peculiar in that its dynamic semantics rely on *runtime type generation*. This choice comes in anticipation of the gradualization of SF, informed by prior work that has used runtime type generation to enforce parametricity dynamically (Matthews and Ahmed, 2008; Ahmed et al., 2011, 2017). The statics of GSF then follow naturally from those of SF using the AGT methodology (§3.3), but the dynamics are more challenging (§3.4/§3.5). In particular, satisfying parametricity forces us to sacrifice the dynamic gradual guarantee in certain scenarios. We study this tension in detail and expose a weaker form of the dynamic gradual guarantee that GSF does satisfy (§3.6). While weaker than the dynamic gradual guarantee as originally intended, this guarantee is stronger than what other gradual parametric languages based on System F achieve; in particular, it implies that imprecise ascriptions are harmless. We then review the notions of gradual parametricity from the literature and present the gradual parametricity that GSF satisfies, along with gradual free theorems (Wadler, 1989)(§3.7). We show that although the notion of parametricity enjoyed by GSF is based on and similar to that of λB , some polymorphic programs behave differently, due to differences in the dynamic semantics of each language.

Supplementary material. Auxiliary definitions and proofs of the main results can be found

in the appendix. Additionally, an interactive prototype of GSF is available, which exhibits both typing derivations and reduction traces, and comes with all the examples mentioned in this work, among others (<https://pleiad.cl/gsf>).

3.1 GSF, Informally

This section presents the design, semantics and metatheory of GSF, a gradual counterpart of System F. Here, we briefly introduce the principles we follow to design GSF, and briefly review its properties (summarized in Table 2.1) and examples of use.

3.1.1 Design Principles

Considering the many concerns involved in developing a gradual language with parametric polymorphism, we should be very clear about the principles, goals and non-goals of a specific design. In designing GSF, we respect the following design principles:

System F syntax: GSF is meant to be a gradual version of System F, and as such, adopts its syntax of both terms and types. Types are only augmented with the unknown type $?$ to introduce the imprecision that is at the core of gradual typing. The design of GSF follows a type-driven approach, precluding the use of unconventional syntactic constructs like the explicit (un)sealing terms of PolyG^ν .

Explicit polymorphism: GSF is a gradual counterpart to System F, and as such, is a fully *explicitly* polymorphic language: type abstraction and type application are part of the term language, reflected in types. GSF gradualizes type information, not term structure.

Simple statics: GSF embodies the complexity of dynamically enforcing parametricity solely in its dynamic semantics; its static semantics is as straightforward as possible.

Natural precision: Precision is intended to capture the level of static typing information of a gradual type, with $?$ as the most imprecise, and static types as the most precise (Siek et al., 2015a). GSF preserves this simple intuition.

3.1.2 Properties

Regarding the challenges and properties discussed previously, here is where GSF stands (Table 2.1):

Type safety: GSF is type safe, meaning all programs either evaluate to a value, halt with a runtime error, or diverge. Well-typed GSF terms do not get stuck.

Parametricity: GSF enforces a notion of gradual parametricity (§3.7), directly inspired by λB (Ahmed et al., 2017).

Conservative extension: GSF is a conservative extension of System F: both languages coincide in their static and dynamic semantics for fully static programs.

Static gradual guarantee: By virtue of the simple statics principle stated above, GSF satisfies the static gradual guarantee, *i.e.* typeability is monotonic with respect to the natural notion of precision.

Dynamic gradual guarantee: GSF does not satisfy the dynamic gradual guarantee (DGG) for the natural notion of precision, but it does satisfy a weaker DGG (§3.6).

Harmless imprecise ascriptions: The weak DGG satisfied by GSF, in particular, implies that imprecise ascriptions are harmless.

Faithful type instantiations: GSF enforces type instantiations of imprecise types.

Expressive imprecision: GSF soundly supports imprecise higher-order polymorphic patterns, bridging the gap towards System F_ω .

Polymorphic interoperability: GSF, like System F and PolyG $^\nu$, only supports explicit polymorphism. This means that certain desirable interoperability scenarios are not supported.

As will be clear by the end of this chapter, the conflict between the DGG and parametricity in a setting that respects the type-driven approach to sealing seems extremely challenging to address, if at all possible. Doing so would require significant changes to the semantics of GSF. In contrast, we believe that the limitation regarding polymorphic interoperability is minor—see illustration and discussion below.

3.1.3 GSF in Action

We now briefly illustrate GSF in action with a number of examples that correspond to the main properties of the language. Other illustrative examples are available with the online interactive prototype. The different sections of the rest of this chapter also come back to such representative examples as needed.

First, System F programs are GSF programs, and behave as expected:

```
let f :  $\forall X.X \rightarrow X = \Lambda X.\lambda x:X.x$  in (f [Int] 1) + 1 ----> 2
```

GSF enforces gradual parametricity. Recall the example from §2.3:

```
let g : ? =  $\lambda a:?.\lambda b:?.\text{if } b \text{ then } a \text{ else } a + 1$  in
let f :  $\forall X.X \rightarrow X = \Lambda X.\lambda x:X.g\ x$   $\boxed{\text{v}}$  in
f [Int] 10
```

As expected, if $\boxed{\text{v}}$ is `true`, the program reduces to 10, and if $\boxed{\text{v}}$ is `false`, the program fails with a runtime error when the body of the function `g` attempts to perform an addition, since this type-specific operation is a violation of parametricity.

In GSF, the natural notion of precision is used for typing, meaning that the following program is a less precise version than the System F program given at the beginning of this section. Also, imprecise ascriptions on values are harmless:

```
let f :  $\forall X.X \rightarrow ?$  =  $\Lambda X.\lambda x:X.x$  in (f [Int] 1) + 1      ----> 2
```

However, as pointed out in the introduction, GSF does not satisfy the dynamic gradual guarantee relative to the natural notion of precision. Consider the following programs:

```
( $\Lambda X.\lambda x:X.x$  :: X) [Int] 1 ----> 1
( $\Lambda X.\lambda x:?.x$  :: X) [Int] 1 ----> error
```

Using the natural notion of term precision, the former is more precise than the latter. Therefore, the dynamic gradual guarantee mandates that less precise term should also reduce to a value, instead of failing. Section 3.6.1 explains the reason for this behavior, after having presented the dynamic semantics of GSF in detail.

GSF enforces type instantiations even when applied to an imprecisely-typed value:

```
let g : ? =  $\Lambda X.\lambda x:X.x$  in g [Int] true      ----> error
```

GSF soundly augments the expressiveness of System F to higher-order polymorphic code:

```
let t : ( $\forall X.?$ )  $\rightarrow ?$  =  $\lambda x:(\forall X.?).x$  [Int] in (t id) 1      ----> 1
let t : ( $\forall X.?$ )  $\rightarrow ?$  =  $\lambda x:(\forall X.?).x$  [Int] in (t id) true ----> error
```

Regarding the limitation of GSF with respect to polymorphic interoperability, the following program fails at runtime:

```
let g : ? =  $\lambda x:(\forall X.X \rightarrow X).x$  [Int] 1
let h : ? =  $\lambda x:?.x$ 
g h
```

The runtime error is raised when `g` is applied to `h`, because `? \rightarrow ?` (the “underlying type” of `h`) is not consistent with the polymorphic function type `$\forall X.X \rightarrow X$` . In certain simple scenarios, it is possible to address this limitation by manually introducing type abstractions or applications, however a more systematic and generally applicable mechanism is definitely desirable. We are studying an extension of GSF with a dynamic adaptation mechanism that addresses polymorphic interoperability. In essence, in the scenario above, the runtime system automatically wraps a type abstraction around `h` instead of failing at the application. A dual adaptation occurs for missing type applications. We conjecture that this mechanism would enable GSF to smoothly support interaction with untyped code, but the full development of this technique is left for future work.

3.2 Preliminary: The static language SF

We systematically derive GSF by applying AGT to a largely standard polymorphic language similar to System F, called SF (Figure 3.1). In addition to the standard System F types and terms, SF includes base types B inhabited by constants b , typed using the auxiliary function

$$\begin{array}{l}
X \in \text{TYPEVAR}, \alpha \in \text{TYPERNAME} \quad \Sigma \in \text{TYPERNAME} \xrightarrow{\text{fin}} \text{TYPE}, \Delta \subset \text{TYPEVAR}, \Gamma \in \text{VAR} \xrightarrow{\text{fin}} \text{TYPE} \\
T ::= B \mid T \rightarrow T \mid \forall X.T \mid T \times T \mid X \mid \alpha \quad (\text{types}) \\
t ::= b \mid \lambda x : T.t \mid \Lambda X.t \mid \langle t, t \rangle \mid x \mid t :: T \mid \text{op}(\bar{t}) \mid t t \mid t [T] \mid \pi_i(t) \quad (\text{terms}) \\
v ::= b \mid \lambda x : T.t \mid \Lambda X.t \mid \langle v, v \rangle \quad (\text{values})
\end{array}$$

$\boxed{\Sigma; \Delta; \Gamma \vdash t : T}$ **Well-typed terms**

$$\begin{array}{c}
\text{(Tb)} \frac{ty(b) = B \quad \Sigma; \Delta \vdash \Gamma}{\Sigma; \Delta; \Gamma \vdash b : B} \qquad \text{(T}\lambda\text{)} \frac{\Sigma; \Delta; \Gamma, x : T \vdash t : T'}{\Sigma; \Delta; \Gamma \vdash \lambda x : T.t : T \rightarrow T'} \\
\text{(T}\Lambda\text{)} \frac{\Sigma; \Delta, X; \Gamma \vdash t : T \quad \Sigma; \Delta \vdash \Gamma}{\Sigma; \Delta; \Gamma \vdash \Lambda X.t : \forall X.T} \qquad \text{(Tpair)} \frac{\Sigma; \Delta; \Gamma \vdash t_1 : T_1 \quad \Sigma; \Delta; \Gamma \vdash t_2 : T_2}{\Sigma; \Delta; \Gamma \vdash \langle t_1, t_2 \rangle : T_1 \times T_2} \\
\text{(Tx)} \frac{x : T \in \Gamma \quad \Sigma; \Delta \vdash \Gamma}{\Sigma; \Delta; \Gamma \vdash x : T} \qquad \text{(Tasc)} \frac{\Sigma; \Delta; \Gamma \vdash t : T \quad \Sigma; \Delta \vdash T = T'}{\Sigma; \Delta; \Gamma \vdash t :: T' : T'} \\
\text{(Top)} \frac{\Sigma; \Delta; \Gamma \vdash \bar{t} : \bar{T}_1 \quad ty(\text{op}) = \bar{T}_2 \rightarrow T}{\Sigma; \Delta \vdash \bar{T}_1 = \bar{T}_2} \qquad \text{(Tapp)} \frac{\Sigma; \Delta; \Gamma \vdash t_1 : T_1 \quad \Sigma; \Delta; \Gamma \vdash t_2 : T_2}{\Sigma; \Delta \vdash \text{dom}(T_1) = T_2} \\
\text{(TappT)} \frac{\Sigma; \Delta; \Gamma \vdash t : T \quad \Sigma; \Delta \vdash T'}{\Sigma; \Delta; \Gamma \vdash t [T'] : \text{inst}(T, T')} \qquad \text{(Tpairi)} \frac{\Sigma; \Delta; \Gamma \vdash t : T}{\Sigma; \Delta; \Gamma \vdash \pi_i(t) : \text{proj}_i(T)} \\
\text{dom} : \text{TYPE} \rightarrow \text{TYPE} \quad \text{cod} : \text{TYPE} \rightarrow \text{TYPE} \quad \text{inst} : \text{TYPE}^2 \rightarrow \text{TYPE} \quad \text{proj}_i : \text{TYPE} \rightarrow \text{TYPE} \\
\text{dom}(T_1 \rightarrow T_2) = T_1 \quad \text{cod}(T_1 \rightarrow T_2) = T_2 \quad \text{inst}(\forall X.T, T') = T[T'/X] \quad \text{proj}_i(T_1 \times T_2) = T_i \\
\text{dom}(T) \text{ undefined o/w} \quad \text{cod}(T) \text{ undefined o/w} \quad \text{inst}(T, T') \text{ undefined o/w} \quad \text{proj}_i(T) \text{ undefined o/w}
\end{array}$$

$\boxed{\Sigma \triangleright t \longrightarrow \Sigma \triangleright t}$ **Notion of reduction**

$$\begin{array}{l}
\Sigma \triangleright v :: T \longrightarrow \Sigma \triangleright v \qquad \Sigma \triangleright \text{op}(\bar{v}) \longrightarrow \Sigma \triangleright \delta(\text{op}, \bar{v}) \qquad \Sigma \triangleright (\lambda x : T.t) v \longrightarrow \Sigma \triangleright t[v/x] \\
\Sigma \triangleright (\Lambda X.t) [T] \longrightarrow \Sigma, \alpha := T \triangleright t[\alpha/X] \quad \text{where } \alpha \notin \text{dom}(\Sigma) \qquad \Sigma \triangleright \pi_i(\langle v_1, v_2 \rangle) \longrightarrow \Sigma \triangleright v_i
\end{array}$$

$\boxed{\Sigma \triangleright t \mapsto \Sigma \triangleright t}$ **Evaluation frames and reduction**

$$\begin{array}{l}
f ::= \square :: T \mid \text{op}(\bar{v}, \square, \bar{t}) \mid \square t \mid v \square \mid \square [T] \mid \langle \square, t \rangle \mid \langle v, \square \rangle \mid \pi_i(\square) \quad (\text{term frames}) \\
\frac{\Sigma \triangleright t \longrightarrow \Sigma' \triangleright t'}{\Sigma \triangleright t \mapsto \Sigma' \triangleright t'} \qquad \frac{\Sigma \triangleright t \mapsto \Sigma' \triangleright t'}{\Sigma \triangleright f[t] \mapsto \Sigma' \triangleright f[t']}
\end{array}$$

Figure 3.1: SF: Static Polymorphic Language with Runtime Type Generation

ty , and primitive n-ary operations op that operate on base types and are given meaning by the function δ . SF also includes pairs $\langle t_1, t_2 \rangle$, and the associated projection operations $\pi_i(t)$,¹ as well as type ascriptions $t :: T$.

The statics are standard. The typing judgment is defined over three contexts: a type name store Σ (explained below), a type variable set Δ that keeps track of type variables in scope, and a standard type environment Γ that associates term variables to types. We adopt the convention of using partial type functions to denote computed types in the rules: dom and cod for domain and codomain types, $inst$ for the resulting type of an instantiation, and $proj_i$ for projected types. These partial functions are undefined if the argument is not of the appropriate shape. We also make the use of type equality explicit as a premise whenever necessary. These conventions are helpful for lifting the static semantics to the gradual setting (Garcia et al., 2016). For closed terms, we write $\cdot; \cdot \vdash t : T$, or simply $\vdash t : T$.

The dynamics are standard call-by-value semantics, specified using reduction frames. The only peculiarity is that they rely on *runtime type generation*. The decision of using type names instead of the traditional substitution semantics is in anticipation of gradualization, and based on prior work that has shown that runtime type generation is key in order to be able to dynamically distinguish between different type variables instantiated with the same type (Matthews and Ahmed, 2008; Ahmed et al., 2011, 2017). We follow the approach already in SF because we want the dynamics and type soundness argument of the static language to help us with GSF, as afforded by AGT (Garcia et al., 2016). Specifically, upon type application, a fresh type name α is generated and bound to the instantiation type T in a global type name store Σ . A type name store Σ maps type names to types; source terms before reduction are typechecked with an empty name store. The notion of reduction and reduction rules all carry along the type name store. While type names only occur at runtime, and not in source programs, reasoning about SF terms as they reduce requires accounting for programs with type names in them. This is why the typing rules are defined relative to a type name store as well. Similarly, type equality (Figure 3.2) is relative to a type name store: a type name α is considered equal to its associated type in the store. The recursive definition of equality modulo type names is necessary to derive equalities (Igarashi et al., 2017a). For instance, in the reduction of the well-typed program $(id \ [Int \rightarrow Int]) \ (id \ [Int])$, where id is the polymorphic identity function, the equality $\alpha := Int \rightarrow Int, \beta := Int; \Delta \vdash \alpha = \beta \rightarrow \beta$ should be derivable.

Rules in Figure 3.1 appeal to auxiliary well-formedness judgments, omitted for brevity. A type T is well-formed $(\Sigma; \Delta \vdash T)$ if it only contains type variables in the type variable environment Δ , and type names bound in a well-formed type name store. A type name store is well-formed $(\vdash \Sigma)$ if all type names are distinct, and associated to types that are well-formed with respect to Σ and the empty type variable environment. A type environment Γ binds term variables to types, and is well-formed $(\Sigma; \Delta \vdash \Gamma)$ if all types are well-formed.

Unsurprisingly, SF is type safe, and all well-typed terms are parametric. These results also follow from the properties of GSF, and the strong relation between both languages.

¹ We omit the constraint $i \in \{1, 2\}$ when operating on pairs throughout this chapter.

$\Sigma; \Delta \vdash T = T$ **Type equality**

$$\begin{array}{c}
\frac{\vdash \Sigma}{\Sigma; \Delta \vdash B = B} \qquad \frac{\vdash \Sigma \quad X \in \Delta}{\Sigma; \Delta \vdash X = X} \qquad \frac{\Sigma; \Delta \vdash T_1 = T'_1 \quad \Sigma; \Delta \vdash T_2 = T'_2}{\Sigma; \Delta \vdash T_1 \rightarrow T_2 = T'_1 \rightarrow T'_2} \\
\\
\frac{\Sigma; \Delta, X \vdash T_1 = T_2}{\Sigma; \Delta \vdash \forall X. T_1 = \forall X. T_2} \qquad \frac{\Sigma; \Delta \vdash T_1 = T'_1 \quad \Sigma; \Delta \vdash T_2 = T'_2}{\Sigma; \Delta \vdash T_1 \times T_2 = T'_1 \times T'_2} \\
\\
\frac{\vdash \Sigma \quad \alpha \in \text{dom}(\Sigma)}{\Sigma; \Delta \vdash \alpha = \alpha} \qquad \frac{\Sigma; \Delta \vdash \Sigma(\alpha) = T}{\Sigma; \Delta \vdash \alpha = T} \qquad \frac{\Sigma; \Delta \vdash T = \Sigma(\alpha)}{\Sigma; \Delta \vdash T = \alpha}
\end{array}$$

Figure 3.2: SF: Type equality

$$\begin{array}{l}
C : \text{GTYPE} \rightarrow \mathcal{P}^*(\text{TYPE}) \\
C(B) = \{B\} \\
C(G_1 \rightarrow G_2) = \{T_1 \rightarrow T_2 \mid T_1 \in C(G_1), T_2 \in C(G_2)\} \\
C(G_1 \times G_2) = \{T_1 \times T_2 \mid T_1 \in C(G_1), T_2 \in C(G_2)\} \\
C(X) = \{X\} \\
C(\alpha) = \{\alpha\} \\
C(\forall X. G) = \{\forall X. T \mid T \in C(G)\} \\
C(?) = \text{TYPE}
\end{array}
\qquad
\begin{array}{l}
A : \mathcal{P}^*(\text{TYPE}) \rightarrow \text{GTYPE} \\
A(\{B\}) = B \\
A(\{\overline{T_{i1}} \rightarrow \overline{T_{i2}}\}) = A(\{\overline{T_{i1}}\}) \rightarrow A(\{\overline{T_{i2}}\}) \\
A(\{\overline{T_{i1}} \times \overline{T_{i2}}\}) = A(\{\overline{T_{i1}}\}) \times A(\{\overline{T_{i2}}\}) \\
A(\{X\}) = X \\
A(\{\alpha\}) = \alpha \\
A(\{\overline{\forall X. T_i}\}) = \forall X. A(\{\overline{T_i}\}) \\
A(\{\overline{T_i}\}) = ? \text{ otherwise}
\end{array}$$

Figure 3.3: Type concretization (C) and abstraction (A)

3.3 GSF: Statics

The first step of the Abstracting Gradual Typing methodology (AGT) is to define the syntax of gradual types and give them meaning through a concretization function to the set of static types they denote. Then, by finding the corresponding abstraction function to establish a Galois connection, the static semantics of the static language can be lifted to the gradual setting.

3.3.1 Syntax and Syntactic Meaning of Gradual Types

We introduce the syntactic category of gradual types $G \in \text{GTYPE}$, by admitting the unknown type in any position, namely:

$$G ::= B \mid G \rightarrow G \mid \forall X. G \mid G \times G \mid X \mid \alpha \mid ?$$

Observe that static types T are syntactically included in gradual types G .

The syntactic meaning of gradual types is straightforward: the unknown type represents any type, and a precise type (constructor) represents the equivalent static type (constructor). For example, $\text{Int} \rightarrow ?$ denotes the set of all function types from Int to any static type. Perhaps surprisingly, we can simply extend this syntactic approach to deal with universal types, type

variables, and type names; the concretization function C is defined in Figure B.5. Note that the definition is purely syntactic and does not even consider well-formedness ($?$ stands for *any* static type); notions built above concretization, such as consistency, will naturally embed the necessary restrictions (§3.3.2). Crucially, choosing to let $?$ stand for any static type means that $?$ can in particular stand for a type variable X (because $X \in C(?)$). Therefore, the gradual type $\forall X. ? \rightarrow X$ includes in its denotation the static types $\forall X. X \rightarrow X$ (the identity function), $\forall X. \text{Int} \rightarrow X$ (a function that always fails when applied), $\forall X. (X * X) \rightarrow X$ (a function that given a pair returns the first or second projection), etc.

Following the abstract interpretation framework, the notion of precision is not subject to tailoring: precision coincides with set inclusion of the denoted static types (Garcia et al., 2016).

Definition 3.1 (Type Precision) $G_1 \sqsubseteq G_2$ if and only if $C(G_1) \subseteq C(G_2)$.

Proposition 3.2 (Precision, inductively) *The inductive definition of type precision given in Figure 3.4 is equivalent to Definition 3.1.*

Observe that both $\forall X. X \rightarrow ?$ and $\forall X. ? \rightarrow X$ are more precise than $\forall X. ? \rightarrow ?$, and less precise than $\forall X. X \rightarrow X$, thereby reflecting the original intuition about precision (Siek and Taha, 2006; Siek et al., 2015a). Also, $\forall X. ? \rightarrow ?$ and $? \rightarrow ?$ are unrelated by precision, since they correspond to different constructors (and GSF is a language with *explicit* polymorphism); they are both more precise than $?$, of course.

Dual to concretization is abstraction, which produces a gradual type from a non-empty set of static types.² The abstraction function A is direct (Figure B.5): it preserves type constructors and falls back on the unknown type whenever a heterogeneous set is abstracted. A is both sound and optimal: it produces the *most precise* gradual type that over-approximates a given set of static types.

Proposition 3.3 (Galois connection) $\langle C, A \rangle$ is a Galois connection, i.e.:

- a) (Soundness) for any non-empty set of static types $S = \{\bar{T}\}$, we have $S \subseteq C(A(S))$
- b) (Optimality) for any gradual type G , we have $A(C(G)) \sqsubseteq G$.

The notion of precision induces a notion of precision *meet* between gradual types, which coincides with the abstraction of the intersection of both concretizations (Garcia et al., 2016).

Definition 3.4 (Precision Meet) $G_1 \sqcap G_2 \triangleq A(C(G_1) \cap C(G_2))$.

Proposition 3.5 (Meet, inductively) *The inductive definition of meet below is equivalent to Definition 3.4.*

²There is no gradual type that denotes an empty set of static types; rather, the empty set corresponds to an error (Garcia et al., 2016).

$$\begin{array}{c}
\frac{}{B \sqcap B = B} \qquad \frac{}{X \sqcap X = X} \qquad \frac{G_1 \sqcap G'_1 = G''_1 \quad G_2 \sqcap G'_2 = G''_2}{(G_1 \rightarrow G_2) \sqcap (G'_1 \rightarrow G'_2) = G''_1 \rightarrow G''_2} \\
\frac{G_1 \sqcap G'_1 = G''_1}{(\forall X.G_1) \sqcap (\forall X.G'_1) = \forall X.G''_1} \qquad \frac{G_1 \sqcap G'_1 = G''_1 \quad G_2 \sqcap G'_2 = G''_2}{(G_1 \times G_2) \sqcap (G'_1 \times G'_2) = G''_1 \times G''_2} \\
\frac{}{\alpha \sqcap \alpha = \alpha} \qquad \frac{}{G \sqcap ? = G} \qquad \frac{}{? \sqcap G = G}
\end{array}$$

3.3.2 Lifting the Static Semantics

The key point of AGT is that once the meaning of gradual types is agreed upon, there is no space for ad hoc design in the static semantics of the language. The abstract interpretation framework provides us with the *definitions* of type predicates and functions over gradual types, for which we can then find equivalent inductive or algorithmic *characterizations*.

In particular, a predicate on static types induces a counterpart on gradual types through *existential* lifting. Our only predicate in SF is type equality, whose existential lifting is type consistency:

Definition 3.6 (Consistency) $\Xi; \Delta \vdash G_1 \sim G_2$ if and only if $\Sigma; \Delta \vdash T_1 = T_2$ for some $\Sigma \in C(\Xi)$, $T_i \in C(G_i)$.

For closed types we write $G_1 \sim G_2$. This definition uses a *gradual* type name store Ξ , which binds type names to gradual types. Its concretization is the pointwise concretization:

$$C(\cdot) = \emptyset \qquad C(\Xi, \alpha := G) = \{ \Sigma, \alpha := T \mid \Sigma \in C(\Xi), T \in C(G) \}$$

Note that because consistency is the consistent lifting of static type equality, which does impose well-formedness, consistency is only defined on well-formed types (*i.e.* $\cdot; \cdot \vdash X \sim X$ does *not* hold).

Proposition 3.7 (Consistency, inductively) *The inductive definition of type consistency given in Figure 3.4 is equivalent to Definition B.96.*

Again, observe that the resulting definition of consistency relates any two types that only differ in unknown type components, without any restriction. Also, because of explicit polymorphism, top-level constructors must match, so $? \rightarrow ?$ is not consistent with $\forall X. ? \rightarrow ?$. However, in line with gradual typing, both are consistent with $?$, as expected. Therefore GSF does not treat $? \rightarrow ?$ as a special “quasi-polymorphic” type, unlike System F_G (Igarashi et al., 2017a). Rather, consistency in GSF coincides with that of CSA (Xie et al., 2018).

Lifting type functions such as *dom* requires abstraction: a lifted function is the abstraction of the results of applying the static function to all the denoted static types (Garcia et al., 2016):

$\alpha \in \text{TYPERNAME} \quad \Xi \in \text{TYPERNAME} \stackrel{\text{fin}}{\text{GTYPE}}, \Delta \subset \text{TYPEVAR}, \Gamma \in \text{VAR} \stackrel{\text{fin}}{\text{GTYPE}}$
 $G ::= B \mid G \rightarrow G \mid \forall X.G \mid G \times G \mid X \mid \alpha \mid ? \quad (\text{gradual types})$
 $t ::= b \mid \lambda x : G.t \mid \Lambda X.t \mid \langle t, t \rangle \mid x \mid t :: G \mid \text{op}(\bar{t}) \mid t t \mid t [G] \mid \pi_i(t) \quad (\text{gradual terms})$

$\Xi; \Delta; \Gamma \vdash t : G$ **Well-typed terms**

$$\begin{array}{c}
(\text{Gb}) \frac{ty(b) = B \quad \Xi; \Delta \vdash \Gamma}{\Xi; \Delta; \Gamma \vdash b : B} \qquad (\text{G}\lambda) \frac{\Xi; \Delta; \Gamma, x : G \vdash t : G'}{\Xi; \Delta; \Gamma \vdash \lambda x : G.t : G \rightarrow G'} \\
(\text{G}\Lambda) \frac{\Xi; \Delta, X; \Gamma \vdash t : G \quad \Xi; \Delta \vdash \Gamma}{\Xi; \Delta; \Gamma \vdash \Lambda X.t : \forall X.G} \qquad (\text{Gpair}) \frac{\Xi; \Delta; \Gamma \vdash t_1 : G_1 \quad \Xi; \Delta; \Gamma \vdash t_2 : G_2}{\Xi; \Delta; \Gamma \vdash \langle t_1, t_2 \rangle : G_1 \times G_2} \\
(\text{G}x) \frac{x : G \in \Gamma \quad \Xi; \Delta \vdash \Gamma}{\Xi; \Delta; \Gamma \vdash x : G} \qquad (\text{Gasc}) \frac{\Xi; \Delta; \Gamma \vdash t : G \quad \Xi; \Delta \vdash G \sim G'}{\Xi; \Delta; \Gamma \vdash t :: G' : G'} \\
(\text{Gop}) \frac{\Xi; \Delta; \Gamma \vdash \bar{t} : \overline{G_1} \quad ty(\text{op}) = \overline{G_2} \rightarrow G}{\Xi; \Delta \vdash \overline{G_1} \sim \overline{G_2}} \qquad (\text{Gapp}) \frac{\Xi; \Delta; \Gamma \vdash t_1 : G_1 \quad \Xi; \Delta; \Gamma \vdash t_2 : G_2}{\Xi; \Delta \vdash \text{dom}^\sharp(G_1) \sim G_2} \\
(\text{GappG}) \frac{\Xi; \Delta; \Gamma \vdash t : G \quad \Xi; \Delta \vdash G'}{\Xi; \Delta; \Gamma \vdash t [G'] : \text{inst}^\sharp(G, G')} \qquad (\text{Gpair}_i) \frac{\Xi; \Delta; \Gamma \vdash t : G}{\Xi; \Delta; \Gamma \vdash \pi_i(t) : \text{proj}_i^\sharp(G)}
\end{array}$$

$\text{dom}^\sharp : \text{GTYPE} \rightarrow \text{GTYPE} \quad \text{cod}^\sharp : \text{GTYPE} \rightarrow \text{GTYPE} \quad \text{inst}^\sharp : \text{GTYPE}^2 \rightarrow \text{GTYPE} \quad \text{proj}_i^\sharp : \text{GTYPE} \rightarrow \text{GTYPE}$
 $\text{dom}^\sharp(G_1 \rightarrow G_2) = G_1 \quad \text{cod}^\sharp(G_1 \rightarrow G_2) = G_2 \quad \text{inst}^\sharp(\forall X.G, G') = G[G'/X] \quad \text{proj}_i^\sharp(G_1 \times G_2) = G_i$
 $\text{dom}^\sharp(?) = ? \quad \text{cod}^\sharp(?) = ? \quad \text{inst}^\sharp(?, G') = ? \quad \text{proj}_i^\sharp(?) = ?$
 $\text{dom}^\sharp(G) \text{ undefined o/w} \quad \text{cod}^\sharp(G) \text{ undefined o/w} \quad \text{inst}^\sharp(G, G') \text{ undefined o/w} \quad \text{proj}_i^\sharp(G) \text{ undefined o/w}$

$\Xi; \Delta \vdash G \sim G$ **Type consistency**

$$\begin{array}{c}
\frac{\vdash \Xi}{\Xi; \Delta \vdash B \sim B} \qquad \frac{\vdash \Xi \quad X \in \Delta}{\Xi; \Delta \vdash X \sim X} \qquad \frac{\Xi; \Delta \vdash G_1 \sim G'_1 \quad \Xi; \Delta \vdash G_2 \sim G'_2}{\Xi; \Delta \vdash G_1 \rightarrow G_2 \sim G'_1 \rightarrow G'_2} \\
\frac{\Xi; \Delta, X \vdash G_1 \sim G_2}{\Xi; \Delta \vdash \forall X.G_1 \sim \forall X.G_2} \qquad \frac{\Xi; \Delta \vdash G_1 \sim G'_1 \quad \Xi; \Delta \vdash G_2 \sim G'_2}{\Xi; \Delta \vdash G_1 \times G_2 \sim G'_1 \times G'_2} \qquad \frac{\vdash \Xi \quad \alpha \in \text{dom}(\Xi)}{\Xi; \Delta \vdash \alpha \sim \alpha} \\
\frac{\Xi; \Delta \vdash \Xi(\alpha) \sim G}{\Xi; \Delta \vdash \alpha \sim G} \qquad \frac{\Xi; \Delta \vdash G \sim \Xi(\alpha)}{\Xi; \Delta \vdash G \sim \alpha} \qquad \frac{\Xi; \Delta \vdash G}{\Xi; \Delta \vdash G \sim ?} \qquad \frac{\Xi; \Delta \vdash G}{\Xi; \Delta \vdash ? \sim G}
\end{array}$$

$G \sqsubseteq G$ **Type precision**

$$\begin{array}{c}
\frac{}{B \sqsubseteq B} \qquad \frac{}{X \sqsubseteq X} \qquad \frac{G_1 \sqsubseteq G'_1 \quad G_2 \sqsubseteq G'_2}{G_1 \rightarrow G_2 \sqsubseteq G'_1 \rightarrow G'_2} \qquad \frac{G_1 \sqsubseteq G_2}{\forall X.G_1 \sqsubseteq \forall X.G_2} \\
\frac{G_1 \sqsubseteq G'_1 \quad G_2 \sqsubseteq G'_2}{G_1 \times G_2 \sqsubseteq G'_1 \times G'_2} \qquad \frac{}{\alpha \sqsubseteq \alpha} \qquad \frac{}{G \sqsubseteq ?}
\end{array}$$

Figure 3.4: GSF: Syntax and Static Semantics

Definition 3.8 (Consistent lifting of functions) *Let F_n be a function of type $\text{TYPE}^n \rightarrow \text{TYPE}$. Its consistent lifting F_n^\sharp , of type $\text{GTYPE}^n \rightarrow \text{GTYPE}$, is defined as: $F_n^\sharp(\overline{G}) = A(\{ F_n(\overline{T}) \mid \overline{T} \in \overline{C(G)} \})$.*

The abstract interpretation framework allows us to prove the following proposition:

Proposition 3.9 (Consistent type functions) *The definitions of dom^\sharp , cod^\sharp , inst^\sharp , and proj_i^\sharp given in Fig. 3.4 are consistent liftings, as per Def. B.97, of the corresponding functions from Fig. 3.1.*

The gradual typing rules of GSF (Figure 3.4) are obtained by replacing type predicates and functions with their corresponding liftings. Note that in (Gapp), the premise $\Xi; \Delta \vdash \text{dom}^\sharp(G_1) \sim G_2$ is a compositional lifting of the corresponding premise in (Tapp), as justified by Garcia et al. (2016). Of particular interest here is the fact that a term of unknown type can be optimistically treated as a polymorphic term and hence instantiated, yielding $?$ as the result type of the type application ($\text{inst}^\sharp(?, G') = ?$). In contrast, a term of function type, even imprecise, cannot be instantiated because the known top-level constructor does not match (*e.g.* $\text{inst}^\sharp(? \rightarrow ?, G')$ is undefined).

3.3.3 Static Properties of GSF

As established by Siek and Taha (2006) in the context of simple types, we can prove that the GSF type system is equivalent to the SF type system on fully-static terms. We say that a gradual type is static if the unknown type does not occur in it, and a term is static if it is fully annotated with static types. Let \vdash_S denote the typing judgment of SF.³

Proposition 3.10 (Static equivalence for static terms) *Let t be a static term and G a static type ($G = T$). We have $\vdash_S t : T$ if and only if $\vdash t : T$.*

The second important property of the static semantics of a gradual language is the static gradual guarantee, which states that typeability is monotonic with respect to precision (Siek et al., 2015a). Type precision (Def. 3.1) extends to *term* precision. A term t is more precise than a term t' if they both have the same structure and t is more precisely annotated than t' . This means that term precision is essentially syntactic, instead of semantic as considered by New et al. (2020). For example, $t \not\sqsubseteq t :: ?$ syntactically, because the two terms do not have the same syntactic structure, but $t :: G \sqsubseteq t :: ?$, where $\vdash t : G$. It is worth pointing out that a semantic precision, such as that established by New et al. (2020), allows easily proving certain desired equivalents between terms in a language that satisfies the DGG. For example, it is desired that the terms t and $t :: ?$ have similar behaviors because a cast or ascription to a less precise type should not affect the behavior of the term t . As we will see in Section 3.6.4, it is possible to prove the equivalence between these terms, t and $t :: ?$, (more generally, between t and $t :: G'$, where $\vdash t :: G$ and $G \sqsubseteq G'$), but using some specific properties of the dynamic semantics. It would be interesting to study a definition of semantic

³As usual, the main propositions are stated over closed terms, but are proven as corollaries of statements over open terms. All statements over open terms can be found in the Appendix.

precision in GSF as future work.

The static gradual guarantee ensures that removing type annotations does not introduce type errors (or dually, that gradual type errors cannot be fixed by making types more precise).

Proposition 3.11 (Static gradual guarantee) *Let t and t' be closed GSF terms such that $t \sqsubseteq t'$ and $\vdash t : G$. Then $\vdash t' : G'$ and $G \sqsubseteq G'$.*

3.4 GSF: Evidence-Based Dynamics

We now turn to the dynamic semantics of GSF. As anticipated, this is where the complexity of gradual parametricity manifests. Still, in addition to streamlining the design of the static semantics, AGT provides effective (though incomplete) guidance for the dynamics. In this section, we first briefly recall the main ingredients of the AGT approach to dynamic semantics, namely *evidence* for consistent judgments and *consistent transitivity*. We then describe the reduction rules of GSF by treating evidence as an abstract datatype. This allows us to clarify a number of key operational aspects before turning in §3.5 to the details of the representation and operations of evidence that enable GSF to satisfy parametricity while adequately tracking type instantiations.

3.4.1 Background: Evidence-Based Semantics for Gradual Languages

For obtaining the dynamic semantics of a gradual language, AGT augments a consistent judgment (such as consistency or consistent subtyping) with the *evidence* of *why* such a judgment holds. Then, reduction mimics proof reduction of the type preservation argument of the static language, combining evidences through steps of *consistent transitivity*, which either yield a more precise evidence, or fail if the evidences to combine are incompatible.⁴ A failure of consistent transitivity corresponds to a cast error in a traditional cast calculus (Garcia et al., 2016).

Consider the gradual typing derivation of $(\lambda x : ?.x+1) \text{ false}$. In the inner typing derivation of the function, the consistent judgment $? \sim \text{Int}$ supports the addition expression, and at the top-level, the judgment $\text{Bool} \sim ?$ supports the application of the function to `false`. When two types are involved in a consistent judgment, we *learn* something about each of these types, namely the justification of *why* the judgment holds. This justification can be captured by a pair of gradual types, $\varepsilon = \langle G_1, G_2 \rangle$, which are at least as precise as the types involved in the judgment (Garcia et al., 2016). (Throughout this chapter, we use the [blue color](#) for evidence ε to enhance readability of the structure of terms.)

⁴In this work, we refer to the evidence of a consistent judgment as a countable entity. Therefore, we use the plural *evidences*, following the accepted use in academic English (Oxford, 2021), instead of writing *pieces of evidence* or *evidence objects*.

$$\varepsilon \Vdash G_1 \sim G_2 \iff \varepsilon \sqsubseteq A^2(\{\langle T_1, T_2 \rangle \mid T_1 \in C(G_1), T_2 \in C(G_2), T_1 = T_2\})$$

$$\text{where } A^2(\{\overline{\langle T_{i1}, T_{i2} \rangle}\}) = \langle A(\overline{\{T_{i1}\}}), A(\overline{\{T_{i2}\}}) \rangle$$

i.e. if evidence $\langle G'_1, G'_2 \rangle$ justifies the consistency judgment $G_1 \sim G_2$, then $G'_1 \sqsubseteq G_1$ and $G'_2 \sqsubseteq G_2$. For instance, by knowing that $? \sim \text{Int}$ holds, we learn that the first type can only possibly be Int , while we do not learn anything new about the right-hand side, which is already fully static. Therefore the evidence of that judgment is $\varepsilon_1 = \langle \text{Int}, \text{Int} \rangle$. Similarly, the evidence for the second judgment is $\varepsilon_2 = \langle \text{Bool}, \text{Bool} \rangle$. Types in evidence can be gradual, *e.g.* $\langle ? \rightarrow ?, ? \rightarrow ? \rangle$ justifies that $(? \rightarrow ?) \sim ?$. Note that with the lifting of simple static type equality, both components of the evidence always coincide, so evidence can be represented as a single gradual type. However, when working with subtyping both components are not necessarily the same, *e.g.* suppose $A <: B$, then $\langle A, B \rangle$ justifies that A is consistent subtype of B . Similarly, type equality in SF is more subtle (§3.2), so the general presentation of evidence as pairs is also required. But for an asymmetric relation such as subtyping, both components are not the same Garcia et al. (2016); *e.g.* suppose $A <: B$, then $\langle A, B \rangle$ justifies that A is consistent subtype of B . Similarly, type equality in SF is more subtle because it is relative to a type name store (§3.2), so the general presentation of evidence as pairs is also required. As an informal example, $\langle \text{Int}, \alpha \rangle$ justifies that $\text{Int} \sim \alpha$ relative to a store in which α is instantiated to Int ; this will be explained in detail in §3.5.1.

At runtime, reduction rules need to *combine* evidences in order to either justify or refute a use of transitivity in the type preservation argument. In our example, we need to combine ε_1 and ε_2 in order to (try to) obtain a justification for the transitive judgment, namely that $\text{Bool} \sim \text{Int}$. The combination operation, called *consistent transitivity* \ddagger , determines whether two evidences support the transitivity: here, $\varepsilon_2 \ddagger \varepsilon_1 = \langle \text{Bool}, \text{Bool} \rangle \ddagger \langle \text{Int}, \text{Int} \rangle$ is undefined, so a runtime error is raised.

The evidence approach is very general and scales to disciplines where consistent judgments are not symmetric, involve more complex reasoning, and even other evidence combination operations (Garcia et al., 2016; Lehmann and Tanter, 2017). All the definitions involved are justified by the abstract interpretation framework. Also, both type safety and the dynamic gradual guarantee become straightforward to prove. In particular, the dynamic gradual guarantee follows directly from the monotonicity (in precision) of consistent transitivity. In fact, the generality of the approach even admits evidence to range over other abstract domains; for instance, for gradual security typing with references, evidence is defined with *label intervals*, not gradual labels (Toro et al., 2018).

3.4.2 Reduction for GSF

In order to denote reduction of (evidence-augmented) gradual typing derivations, Garcia et al. (2016) use *intrinsic* terms as a notational device; while appropriate, the resulting description is fairly hard to comprehend and unusual, and it does implicitly involve a (presentational) transformation from source terms to their intrinsic representation. In this work, we simplify the exposition by avoiding the use of intrinsic terms; instead, we rely on a type-directed,

$t ::= v \mid \langle t, t \rangle \mid x \mid \varepsilon t :: G \mid op(\bar{t}) \mid t t \mid t [G] \mid \pi_i(t)$ (terms)
 $v ::= \varepsilon u :: G$ (values)
 $u ::= b \mid \lambda x : G.t \mid \Lambda X.t \mid \langle u, u \rangle$ (raw values)

$\Xi; \Delta; \Gamma \vdash s : G$ **Well-typed terms** (for conciseness, s ranges over both t and u)

$$\begin{array}{c}
 (Eb) \frac{ty(b) = B \quad \Xi; \Delta \vdash \Gamma}{\Xi; \Delta; \Gamma \vdash b : B} \qquad (E\lambda) \frac{\Xi; \Delta; \Gamma, x : G \vdash t : G'}{\Xi; \Delta; \Gamma \vdash \lambda x : G.t : G \rightarrow G'} \\
 (E\Lambda) \frac{\Xi; \Delta, X; \Gamma \vdash t : G \quad \Xi; \Delta \vdash \Gamma}{\Xi; \Delta; \Gamma \vdash \Lambda X.t : \forall X.G} \qquad (E_{\text{pair}}) \frac{\Xi; \Delta; \Gamma \vdash s_1 : G_1 \quad \Xi; \Delta; \Gamma \vdash s_2 : G_2}{\Xi; \Delta; \Gamma \vdash \langle s_1, s_2 \rangle : G_1 \times G_2} \\
 (Ex) \frac{x : G \in \Gamma \quad \Xi; \Delta \vdash \Gamma}{\Xi; \Delta; \Gamma \vdash x : G} \qquad (E_{\text{asc}}) \frac{\Xi; \Delta; \Gamma \vdash s : G \quad \varepsilon \Vdash \Xi; \Delta \vdash G \sim G'}{\Xi; \Delta; \Gamma \vdash \varepsilon s :: G' : G'} \\
 (E_{\text{op}}) \frac{\Xi; \Delta; \Gamma \vdash \bar{t} : \bar{G} \quad ty(op) = \bar{G} \rightarrow G'}{\Xi; \Delta; \Gamma \vdash op(\bar{t}) : G'} \qquad (E_{\text{app}}) \frac{\Xi; \Delta; \Gamma \vdash t_1 : G \rightarrow G' \quad \Xi; \Delta; \Gamma \vdash t_2 : G}{\Xi; \Delta; \Gamma \vdash t_1 t_2 : G'} \\
 (E_{\text{app}G}) \frac{\Xi; \Delta; \Gamma \vdash t : \forall X.G \quad \Xi; \Delta \vdash G'}{\Xi; \Delta; \Gamma \vdash t [G'] : G[G'/X]} \qquad (E_{\text{pair}i}) \frac{\Xi; \Delta; \Gamma \vdash t : G_1 \times G_2}{\Xi; \Delta; \Gamma \vdash \pi_i(t) : G_i}
 \end{array}$$

$\Xi \triangleright t \longrightarrow \Xi \triangleright t$ or **error** **Notion of reduction**

$$\begin{array}{c}
 (R_{\text{asc}}) \quad \Xi \triangleright \varepsilon_2(\varepsilon_1 u :: G_1) :: G_2 \longrightarrow \begin{cases} \Xi \triangleright (\varepsilon_1 \ ; \ \varepsilon_2)u :: G_2 \\ \mathbf{error} & \text{if not defined} \end{cases} \\
 (R_{\text{op}}) \quad \Xi \triangleright op(\overline{\varepsilon u} :: \bar{G}) \longrightarrow \Xi \triangleright \varepsilon_B \delta(op, \bar{u}) :: B \quad \text{where } B \triangleq cod(ty(op)) \\
 (R_{\text{app}}) \quad \Xi \triangleright (\varepsilon_1(\lambda x : G_{11}.t) :: G_1 \rightarrow G_2) (\varepsilon_2 u :: G_1) \longrightarrow \begin{cases} \Xi \triangleright cod(\varepsilon_1)(t[(\varepsilon_2 \ ; \ dom(\varepsilon_1))u :: G_{11}]/x) :: G_2 \\ \mathbf{error} & \text{if not defined} \end{cases} \\
 (R_{\text{pair}}) \quad \Xi \triangleright \langle \varepsilon_1 u_1 :: G_1, \varepsilon_2 u_2 :: G_2 \rangle \longrightarrow \Xi \triangleright (\varepsilon_1 \times \varepsilon_2) \langle u_1, u_2 \rangle :: G_1 \times G_2 \\
 (R_{\text{proj}i}) \quad \Xi \triangleright \pi_i(\varepsilon \langle u_1, u_2 \rangle :: G_1 \times G_2) \longrightarrow \Xi \triangleright p_i(\varepsilon) u_i :: G_i \\
 (R_{\text{app}G}) \quad \Xi \triangleright (\varepsilon \Lambda X.t :: \forall X.G) [G'] \longrightarrow \Xi' \triangleright \varepsilon_{out}(\varepsilon[\hat{\alpha}]t[\hat{\alpha}/X] :: G[\alpha/X]) :: G[G'/X] \\
 \text{where } \Xi' \triangleq \Xi, \alpha := G' \text{ for some } \alpha \notin dom(\Xi) \\
 \text{and } \hat{\alpha} = lift_{\Xi'}(\alpha)
 \end{array}$$

$\Xi \triangleright t \mapsto \Xi \triangleright t$ or **error** **Evaluation frames and reduction**

$$\begin{array}{c}
 f ::= \varepsilon \square :: G \mid op(\bar{v}, \square, \bar{t}) \mid \square t \mid v \square \mid \square [G] \mid \langle \square, t \rangle \mid \langle v, \square \rangle \mid \pi_i(\square) \\
 (R \longrightarrow) \frac{\Xi \triangleright t \longrightarrow \Xi' \triangleright t'}{\Xi \triangleright t \mapsto \Xi' \triangleright t'} \qquad (Rf) \frac{\Xi \triangleright t \mapsto \Xi' \triangleright t'}{\Xi \triangleright f[t] \mapsto \Xi' \triangleright f[t']} \\
 (R_{\text{err}}) \frac{\Xi \triangleright t \longrightarrow \mathbf{error}}{\Xi \triangleright t \mapsto \mathbf{error}} \qquad (Rf_{\text{err}}) \frac{\Xi \triangleright t \mapsto \mathbf{error}}{\Xi \triangleright f[t] \mapsto \mathbf{error}}
 \end{array}$$

Figure 3.5: GSF ε : Syntax, Static and Dynamic Semantics

straightforward translation to $\text{GSF}\varepsilon$, a simple variant of GSF in which all values are ascribed, and ascriptions carry evidence. The translation, described formally below (§3.4.3), inserts explicit ascriptions everywhere consistency is used—very much in the spirit of the coercion-based semantics of subtyping (Pierce, 2002).

For instance, the small program of §3.4.1 above, $(\lambda x : ?.x + 1) \text{ false}$, is translated to:

$$(\varepsilon_{? \rightarrow \text{Int}}(\lambda x : ?.(\varepsilon_1 x :: \text{Int}) + (\varepsilon_{\text{Int}} 1 :: \text{Int})) :: ? \rightarrow \text{Int}) (\varepsilon_2(\varepsilon_{\text{Bool}} \text{false} :: \text{Bool}) :: ?)$$

where ε_G is the evidence of the reflexive judgment $G \sim G$ (e.g. ε_{Int} supports $\text{Int} \sim \text{Int}$). Evidences $\varepsilon_1 = \langle \text{Int}, \text{Int} \rangle$ and $\varepsilon_2 = \langle \text{Bool}, \text{Bool} \rangle$ are the ones from §3.4.1. Recall that ε_1 is evidence of the consistency judgment $? \sim \text{Int}$, where $?$ is the type of x , and Int comes from the ascription; likewise ε_2 is evidence of the consistency judgment $\text{Bool} \sim ?$. Such initial evidences are computed by means of an *interior* function \mathcal{I} , given by the abstract interpretation framework (Garcia et al., 2016): in this setting, the interior coincides with the precision meet (§3.3.1), i.e. $\mathcal{I}(G_1, G_2) = \langle G_1 \sqcap G_2, G_1 \sqcap G_2 \rangle$.

This translation preserves the essence of the AGT dynamics approach in which evidence and consistent transitivity drive the runtime monitoring aspect of gradual typing. Furthermore, by making the translation explicitly ascribe all base values to their base type, $\text{GSF}\varepsilon$ can feature a uniform syntax and greatly simplified reduction rules, compared to the original AGT exposition. This presentation also streamlines the proofs by reducing the number of cases to consider.

Figure 3.5 presents the syntax and semantics of $\text{GSF}\varepsilon$, a simple variant of GSF in which all values are ascribed, and ascriptions carry evidence. Key changes with respect to Figure 3.4 are highlighted in gray. Here, we treat evidence as a pair of elements of an *abstract* datatype; we define its actual representation (and operations) in the next section.

As we will see in Section 3.4.3, the translation from GSF to $\text{GSF}\varepsilon$ introduces explicit ascriptions everywhere consistency is used, leaving rule (*Easc*) as the only remaining use of consistency in the typing rules. The evidence of the term itself supports the consistency judgment in the premise. All other rules require types to match exactly; the translation inserts ascriptions to ensure that top-level constructors match in every elimination form.

The notion of reduction for $\text{GSF}\varepsilon$ terms deals with evidence propagation and composition with consistent transitivity. Rule (*Rasc*) specifies how an ascription around an ascribed value reduces to a single value if consistent transitivity holds: ε_1 justifies that $G_u \sim G_1$, where G_u is the type of the underlying simple value u , and ε_2 is evidence that $G_1 \sim G_2$. The composition via consistent transitivity $\mathbin{\text{;}}$ justifies that $G_u \sim G_2$; if the composition is undefined, reduction steps to **error**. Rule (*Rop*) simply strips the underlying simple values, applies the primitive operation, and then wraps the result in an ascription, using a canonical base evidence ε_B (which trivially justifies that $B \sim B$). Rule (*Rapp*) combines the evidence from the argument value ε_2 with the domain evidence of the function value $\text{dom}(\varepsilon_1)$ in an attempt to transitively justify that $G_u \sim G_{11}$. Failure to justify that judgment, as in our example in §3.4.1, produces **error**. The return value is ascribed to the expected return type, using the codomain evidence of the function $\text{cod}(\varepsilon_1)$. Rule (*Rpair*) produces a pair value when the subterms of a pair have been reduced to values themselves, using the product operator on evidences $\varepsilon_1 \times \varepsilon_2$. This rule is necessary to enforce a uniform presentation of all values as ascribed values, which simplifies

technicalities. Dually, Rule (*Rproj_i*) extracts a component of a pair and ascribes it to the projected type, using the corresponding evidence obtained with $p_i(\varepsilon)$ (not to be confused with $\pi_i(\varepsilon)$, which refers to the first or second projection of evidence, itself a metalanguage pair).

Apart from the presentational details, the above rules are standard for an evidence-based reduction semantics. Rule (*RappG*) is *the* rule that specifically deals with parametric polymorphism, reducing a type application.

$$\Xi \triangleright (\varepsilon \Lambda X.t :: \forall X.G) [G'] \longrightarrow \Xi' \triangleright \varepsilon_{out}(\varepsilon[\hat{\alpha}]t[\hat{\alpha}/X] :: G[\alpha/X]) :: G[G'/X]$$

where $\Xi' \triangleq \Xi, \alpha := G'$ for some $\alpha \notin \text{dom}(\Xi)$ and $\hat{\alpha} = \text{lift}_{\Xi'}(\alpha)$

This is where most of the complexity of gradual parametricity concentrates. Observe that there are two ascriptions in the produced term:

- The *inner* ascription (to $G[\alpha/X]$) is for the body of the polymorphic term, asserting that substituting a fresh type name α for the type variable X preserves typing. The associated evidence $\varepsilon[\hat{\alpha}]$ is the result of instantiating ε (which justifies that the actual type of $\Lambda X.t$ is consistent with $\forall X.G$) with the fresh type name, hence justifying that the body after substitution is consistent with $G[\alpha/X]$. The operator $\text{lift}_{\Xi'}(\alpha)$, and substitution operations $t[\hat{\alpha}/X]$ and $\varepsilon[\hat{\alpha}]$ are left abstract for now (as evidence is abstract) and defined later in §3.5.2.
- The *outer* ascription asserts that $G[\alpha/X]$ is consistent with $G[G'/X]$, witnessed by evidence ε_{out} . We define ε_{out} in §3.5.2 below, once the representation of evidence is introduced.

Instead of using these two evidences, we could have used directly their composition, $\varepsilon[\hat{\alpha}] \circledast \varepsilon_{out}$. But the approach used here makes the definition of the logical relation clearer and the proofs easier.

The use of $\hat{\alpha}$ is a technicality: because so far we treat evidence as an abstract datatype from an as-yet-unspecified domain, say pairs of ETYPE, we cannot directly use gradual types (GTYPE) inside evidences. The connection between GTYPE and ETYPE is specified by lifting operators, $\text{lift}_{\Xi} : \text{GTYPE} \rightarrow \text{ETYPE}$ and $\text{unlift} : \text{ETYPE} \rightarrow \text{GTYPE}$.⁵ We define these operators later (Figure 3.7), after the structure of evidences has been explained in detail. Because type names have meaning related to a store, the lifting is parameterized by the store Ξ . Type substitution in terms is mostly standard: it uses *unlift* to recover α , and is extended to substitute recursively in evidences. Substitution in evidence, also triggered by evidence instantiation, is simply component-wise substitution on evidence types. Both substitution operators are formally defined later (Figure 3.8).

Finally, the evaluation frames and associated reduction rules in Figure 3.5 are straightforward; in particular (*Rerr*) and (*Rferr*) propagate **error** to the top-level.

$\Delta; \Gamma \vdash v \rightsquigarrow_v u : G$ **Value translation**

$$\begin{array}{c}
\text{(Gb)} \frac{ty(b) = B \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash b \rightsquigarrow_v b : B} \qquad \text{(Gpairu)} \frac{\Delta; \Gamma \vdash v_1 \rightsquigarrow u_1 : G_1 \quad \Delta; \Gamma \vdash v_2 \rightsquigarrow u_2 : G_2}{\Delta; \Gamma \vdash \langle v_1, v_2 \rangle \rightsquigarrow_v \langle u_1, u_2 \rangle : G_1 \times G_2} \\
\text{(G}\lambda\text{)} \frac{\Delta; \Gamma, x : G \vdash t \rightsquigarrow t' : G'}{\Delta; \Gamma \vdash (\lambda x : G.t) \rightsquigarrow_v (\lambda x : G.t') : G \rightarrow G'} \qquad \text{(G}\Lambda\text{)} \frac{\Delta, X; \Gamma \vdash t \rightsquigarrow t' : G \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash (\Lambda X.t) \rightsquigarrow_v (\Lambda X.t') : \forall X.G}
\end{array}$$

$\Delta; \Gamma \vdash t \rightsquigarrow t : G$ **Term translation**

$$\begin{array}{c}
\text{(Gu)} \frac{\Delta; \Gamma \vdash v \rightsquigarrow_v u : G \quad \varepsilon = \mathcal{I}(G, G')}{\Delta; \Gamma \vdash v \rightsquigarrow \varepsilon u :: G : G} \qquad \text{(Gascu)} \frac{\Delta; \Gamma \vdash v \rightsquigarrow_v u : G \quad \varepsilon = \mathcal{I}(G, G')}{\Delta; \Gamma \vdash v :: G' \rightsquigarrow \varepsilon u :: G' : G'} \\
\text{(Gx)} \frac{x : G \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x \rightsquigarrow x : G} \qquad \text{(Gasct)} \frac{t \neq v \quad \Delta; \Gamma \vdash t \rightsquigarrow t' : G \quad \varepsilon = \mathcal{I}(G, G')}{\Delta; \Gamma \vdash t :: G' \rightsquigarrow \varepsilon t' :: G' : G'} \\
\text{(Gpairt)} \frac{(t_1 \neq v_1 \vee t_2 \neq v_2) \quad \Delta; \Gamma \vdash t_1 \rightsquigarrow t'_1 : G_1 \quad \Delta; \Gamma \vdash t_2 \rightsquigarrow t'_2 : G_2}{\Delta; \Gamma \vdash \langle t_1, t_2 \rangle \rightsquigarrow \langle t'_1, t'_2 \rangle : G_1 \times G_2} \\
\text{(Gop)} \frac{\Delta; \Gamma \vdash \bar{t} \rightsquigarrow \bar{t}' : \overline{G_1} \quad ty(op) = \overline{G_2} \rightarrow G \quad \bar{\varepsilon} = \overline{\mathcal{I}(G_1, G_2)}}{\Delta; \Gamma \vdash op(\bar{t}) \rightsquigarrow op(\bar{\varepsilon} t' :: G_2) : G} \\
\text{(Gapp)} \frac{\Delta; \Gamma \vdash t_1 \rightsquigarrow t'_1 : G_1 \quad G_1 \rightarrow G'_1 \rightarrow G'_2 \quad \varepsilon_1 = \mathcal{I}(G_1, G'_1 \rightarrow G'_2) \quad \varepsilon_2 = \mathcal{I}(G_2, G'_1)}{\Delta; \Gamma \vdash t_2 \rightsquigarrow t'_2 : G_2} \\
\Delta; \Gamma \vdash t_1 t_2 \rightsquigarrow (\varepsilon_1 t_1 :: G'_1 \rightarrow G'_2) (\varepsilon_2 t_2 :: G'_1) : G'_2 \\
\text{(GappG)} \frac{\Delta; \Gamma \vdash t \rightsquigarrow t' : G \quad G \rightarrow \forall X.G'' \quad \Delta \vdash G' \quad \varepsilon = \mathcal{I}(G, \forall X.G'')}{\Delta; \Gamma \vdash t [G'] \rightsquigarrow (\varepsilon t' :: \forall X.G'') [G'] : G''[G'/X]} \\
\text{(Gpairi)} \frac{\Delta; \Gamma \vdash t \rightsquigarrow t' : G \quad G \rightarrow G_1 \times G_2 \quad \varepsilon = \mathcal{I}(G, G_1 \times G_2)}{\Delta; \Gamma \vdash \pi_i(t) \rightsquigarrow \pi_i(\varepsilon t' :: G_i) : G_i}
\end{array}$$

$G \rightarrow G$ **Type matching**

$$\begin{array}{c}
G_1 \rightarrow G_2 \rightarrow G_1 \rightarrow G_2 \quad \forall X.G \rightarrow \forall X.G \quad G_1 \times G_2 \rightarrow G_1 \times G_2 \quad ? \rightarrow ? \rightarrow ? \quad ? \rightarrow \forall X.? \\
? \rightarrow ? \times ?
\end{array}$$

Figure 3.6: Translation from GSF to GSF ε

3.4.3 Elaborating GSF to GSF ε

Figure 3.6 defines the type-preserving translation from GSF to GSF ε by using two mutually-defined translations: \rightsquigarrow translates GSF terms to GSF ε terms, and \rightsquigarrow_v translates GSF values to GSF ε raw values. Raw values are treated separately as they are not values (and thus not terms) and must be ascribed upon translation. The translation follows naturally from the typing rules of GSF. We use metavariable v in GSF to range over constants, functions, type

⁵ In standard AGT (Garcia et al., 2016) the lifting is simply the identity, *i.e.* ETYPE = GTYPE.

abstractions and pairs of v , and use \rightsquigarrow_v to translate them to raw values u . Rule (Gu) and (Gascu) translate a value v and an ascribed value in GSF, respectively, to a $\text{GSF}\varepsilon$ value, using \rightsquigarrow_v , producing a new ascribed raw value, *i.e.* a value. Note that we could have translated GSF values v to $\text{GSF}\varepsilon$ values directly but it would have generated redundant ascriptions such as $b :: \text{Bool} \rightsquigarrow \varepsilon_{\text{Bool}}(\varepsilon_{\text{Bool}}b :: \text{Bool}) :: \text{Bool}$. Note that these rules use the interior \mathcal{G} to calculate the initial evidence. The rule (Gasct) is similar to (Gascu), but it uses \rightsquigarrow to translate GSF terms that are not values. Rules (Gapp), (GappG) and (Gpairi) use *type matching* \rightarrow (Cimini and Siek, 2016) to ascribe subterms of type $?$ in elimination positions to the corresponding top-level type constructor, *e.g.* $\forall X.?$ for a type application, and $? \rightarrow ?$ for a function application. For subterms of a more precise type, type matching is the identity. Note that in (Gapp), the argument is also ascribed to the type of the domain G'_1 obtained during type matching.

We can show straightforwardly by induction on typing judgments that the translation preserves typing (Lemma 3.12).

Lemma 3.12 (Translation Preserves Typing) *Let t be a GSF term. If $\Delta; \Gamma \vdash t : G$ then $\Delta; \Gamma \vdash t \rightsquigarrow t_\varepsilon : G$ and $\Delta; \Gamma \vdash t_\varepsilon : G$.*

3.5 GSF: Evidence for Gradual Parametricity

As highlighted in Section 3.4, AGT provides effective (though incomplete) guidance for the dynamics. The dynamic semantics obtained by applying AGT ensure type safety, but unfortunately not parametricity. Ensuring parametricity requires a refined representation of evidence and definition of consistent transitivity. This can be considered as a shortcoming of the AGT methodology, already observed in the context of security typing (Toro et al., 2018): some properties of the static language may not be preserved by gradualization. We first explain in §3.5.1 why the standard representation of evidence as pair of gradual types is insufficient for gradual parametricity. We then introduce the refined representation of evidence to enforce parametricity (§3.5.2), and basic properties of the language. Richer properties of GSF are discussed in §3.6 and §3.7.

3.5.1 Simple Evidence, and Why It Fails

In standard AGT (Garcia et al., 2016), evidence is simply represented as a pair of gradual types: an evidence ε is of the form $\langle G_1, G_2 \rangle$. The two constituents of an evidence are not necessarily the same, *e.g.* when considering non-symmetric judgments such as subtyping. Consistent transitivity is defined through the abstract interpretation framework. Write $\varepsilon \Vdash J$ to denote that ε justifies the consistent judgment J . The definition of consistent transitivity for simple types is as follows:

Definition 3.13 (Consistent transitivity for simple type equality (Garcia et al., 2016)) *Suppose $\varepsilon_{ab} \Vdash G_a \sim G_b$ and $\varepsilon_{bc} \Vdash G_b \sim G_c$. Evidence for consistent transitivity is deduced as*

$(\varepsilon_{ab} \circledast \varepsilon_{bc}) \Vdash G_a \sim G_c$, where:

$$\langle G_1, G_{21} \rangle \circledast \langle G_{22}, G_3 \rangle = A^2(\{\langle T_1, T_3 \rangle \in C(G_1) \times C(G_3) \mid \exists T_2 \in C(G_{21}) \cap C(G_{22}), T_1 = T_2 \wedge T_2 = T_3\})$$

In words, if defined, the evidence that supports the transitive judgment is obtained by abstracting over the pairs of static types denoted by the outer evidence types (G_1 and G_3) *such that* they are connected through a static type common to both middle evidence types (G_{21} and G_{22}). Note that for consistent transitivity to be defined, $C(G_{21}) \cap C(G_{22})$ must not be empty. This definition can be proven to be equivalent to an inductive definition that proceeds in a syntax-directed manner on the structure of types (Garcia et al., 2016).

Consistent transitivity satisfies some important properties. First, it is associative. Second, the resulting evidence is more precise than the outer evidence types, reflecting that during evaluation, typing justification only gets more precise (or fails). Violating this property breaks type safety. Third, it is monotonic; this property is key for establishing the dynamic gradual guarantee (Garcia et al., 2016). Here, an evidence ε is more precise than ε' , written $\varepsilon \sqsubseteq \varepsilon'$, if $\pi_1(\varepsilon) \sqsubseteq \pi_1(\varepsilon')$ and $\pi_2(\varepsilon) \sqsubseteq \pi_2(\varepsilon')$.

Lemma 3.14 (*Properties of consistent transitivity*).

- (a) *Associativity.* $(\varepsilon_1 \circledast \varepsilon_2) \circledast \varepsilon_3 = \varepsilon_1 \circledast (\varepsilon_2 \circledast \varepsilon_3)$, or both are undefined.
- (b) *Optimality.* If $\varepsilon = \varepsilon_1 \circledast \varepsilon_2$ is defined, then $\pi_1(\varepsilon) \sqsubseteq \pi_1(\varepsilon_1)$ and $\pi_2(\varepsilon) \sqsubseteq \pi_2(\varepsilon_2)$.
- (c) *Monotonicity.* If $\varepsilon_1 \sqsubseteq \varepsilon'_1$ and $\varepsilon_2 \sqsubseteq \varepsilon'_2$ and $\varepsilon_1 \circledast \varepsilon_2$ is defined, then $\varepsilon'_1 \circledast \varepsilon'_2$ is defined and $\varepsilon_1 \circledast \varepsilon_2 \sqsubseteq \varepsilon'_1 \circledast \varepsilon'_2$.

Unfortunately, systematically following the AGT methodology and simply extending the consistent transitivity definition to deal with GSF types and consistency judgments yields a gradual language that breaks parametricity.⁶ Let us first adapt the simple definition of consistent transitivity (Def. 3.13) to the GSF consistency judgment, which is stated relative to type names and type variables environments:

Definition 3.15 (Consistent transitivity for GSF—simple attempt) *Suppose $\varepsilon_{ab} \Vdash \Xi; \Delta \vdash G_a \sim G_b$ and $\varepsilon_{bc} \Vdash \Xi; \Delta \vdash G_b \sim G_c$. Evidence for consistent transitivity is deduced as $(\varepsilon_{ab} \circledast \varepsilon_{bc}) \Vdash \Xi; \Delta \vdash G_a \sim G_c$, where:*

$$\begin{aligned} \langle G_1, G_{21} \rangle \circledast \langle G_{22}, G_3 \rangle = & A^2(\{\langle T_1, T_3 \rangle \in C(G_1) \times C(G_3) \mid \\ & \exists T_2 \in C(G_{21}) \cap C(G_{22}) \wedge \Sigma \in C(\Xi) \wedge \\ & \Sigma; \Delta \vdash T_1 = T_2 \wedge \Sigma; \Delta \vdash T_2 = T_3\}) \end{aligned}$$

where Σ is the static counterpart of Ξ , *i.e.* a mapping from type names to static types (§3.3.2).

As previously mentioned, type equality in SF (Figure 3.2) is more subtle than the simple static type equality. The general representation of evidence as pairs is required (as opposition to use evidence as a single type) because each type in the evidence corresponds to each type in the judgment, which can be different. For example, suppose that α is equal to Int in the

⁶The obtained language is type safe, and satisfies the dynamic gradual guarantee. This novel design could make sense to gradualize impure polymorphic languages, which do not enforce parametricity.

store $(\Xi = \alpha := \text{Int})$. Then $\langle \text{Int}, \alpha \rangle$ is evidence that Int is consistent with α , given Ξ , *i.e.*

$$\langle \text{Int}, \alpha \rangle \Vdash \Xi; \cdot \vdash \text{Int} \sim \alpha$$

Evidence as a pair of types is crucial in the representation of the outer evidence ε_{out} during reduction. Remember that if a type abstraction with type $\forall X.G$ is applied to G' , the resulting ε_{out} justifies that $G[\alpha/X]$ is consistent with $G[G'/X]$, where α is the generated fresh type name. Informally, if $\varepsilon_{out} \Vdash \Xi, \alpha := G' \vdash G[\alpha/X] \sim G[G'/X]$, then ε_{out} is computed as $\langle G[\alpha/X], G[G'/X] \rangle$. Thus, if a type abstraction with type $\forall X.X \rightarrow X$ is applied to Int , ε_{out} is computed as $\langle \alpha \rightarrow \alpha, \text{Int} \rightarrow \text{Int} \rangle$, where ε_{out} justifies that $(X \rightarrow X)[\alpha/X] = \alpha \rightarrow \alpha$ is consistent with $(X \rightarrow X)[\text{Int}/X] = \text{Int} \rightarrow \text{Int}$. An evidence such as $\langle \text{Int}, \alpha \rangle$ can be obtained from the domain information of ε_{out} , namely $dom(\varepsilon_{out})$. Taking this definition of ε_{out} into account, we can now illustrate the problem of the consistent transitivity definition derived by the AGT methodology. Consider the following simple program:

```
1  (ΛX.(λx:X.(x :: ? :: ?) + 1)) [Int] 1
```

The function above is not parametric because it ends up adding 1 to its argument—although it does so after two intermediate ascriptions to the type $?$. Without further precaution, the parametricity violation of this program would not be detected at runtime. Note that two ascriptions are needed in order to elaborate the evidence $\langle ?, ? \rangle$, used below to illustrate the problem. The following reduction trace illustrates all the important aspects of reduction (assuming that the type application in the program below generates the fresh name α , bound to Int in the store):

$$\begin{aligned}
& \cdot \triangleright (\varepsilon_{\forall X.X \rightarrow \text{Int}}(\Lambda X.\lambda x : X.(\varepsilon_?(\varepsilon_X x :: ?) :: ?) + \varepsilon_{\text{Int}}1 :: \text{Int}) :: \forall X.X \rightarrow \text{Int}) [\text{Int}] (\varepsilon_{\text{Int}}1 :: \text{Int}) \\
\mapsto^* \alpha := \text{Int} & \triangleright \varepsilon_{out}(\varepsilon_{\alpha \rightarrow \alpha}(\lambda x : \alpha.(\varepsilon_?(\varepsilon_\alpha x :: ?) :: ?) + \varepsilon_{\text{Int}}1 :: \text{Int}) :: \alpha \rightarrow \alpha) :: \text{Int} \rightarrow \text{Int} (\varepsilon_{\text{Int}}1 :: \text{Int}) \\
\mapsto^* \alpha := \text{Int} & \triangleright \varepsilon_{out}(\lambda x : \alpha.(\varepsilon_?(\varepsilon_\alpha x :: ?) :: ?) + \varepsilon_{\text{Int}}1 :: \text{Int}) :: \text{Int} \rightarrow \text{Int} (\varepsilon_{\text{Int}}1 :: \text{Int}) \\
\mapsto^* \alpha := \text{Int} & \triangleright cod(\varepsilon_{out})((\varepsilon_?(\varepsilon_\alpha((\varepsilon_{\text{Int}} \text{;} dom(\varepsilon_{out}))1 :: \alpha) :: ?) :: ?) + \varepsilon_{\text{Int}}1 :: \text{Int}) :: \text{Int} \\
\text{But } dom(\varepsilon_{out}) &= \langle \text{Int}, \alpha \rangle \text{ and } cod(\varepsilon_{out}) = \langle \text{Int}, \text{Int} \rangle \\
= \alpha := \text{Int} & \triangleright \langle \text{Int}, \text{Int} \rangle((\varepsilon_?(\varepsilon_\alpha(\langle \text{Int}, \alpha \rangle 1 :: \alpha) :: ?) :: ?) + \varepsilon_{\text{Int}}1 :: \text{Int}) :: \text{Int} \\
\mapsto^* \alpha := \text{Int} & \triangleright \langle \text{Int}, \text{Int} \rangle((\varepsilon_?(\langle \text{Int}, \alpha \rangle 1 :: ?) :: ?) + \varepsilon_{\text{Int}}1 :: \text{Int}) :: \text{Int} \\
\mapsto^* \alpha := \text{Int} & \triangleright \langle \text{Int}, \text{Int} \rangle(\langle \text{Int}, ? \rangle 1 :: ? + \varepsilon_{\text{Int}}1 :: \text{Int}) :: \text{Int} \\
\mapsto^* \alpha := \text{Int} & \triangleright \langle \text{Int}, \text{Int} \rangle(\langle \text{Int}, \text{Int} \rangle 2 :: \text{Int}) :: \text{Int} \\
\mapsto^* \alpha := \text{Int} & \triangleright \langle \text{Int}, \text{Int} \rangle 2 :: \text{Int}
\end{aligned}$$

The summary below illustrates the main evidences that arise in the above reduction, showing the judgment they justify in each case:

$$\begin{aligned}
\varepsilon_{\forall X.X \rightarrow \text{Int}} &= \langle \forall X.X \rightarrow \text{Int}, \forall X.X \rightarrow \text{Int} \rangle \Vdash \cdot; \cdot \quad \vdash \forall X.X \rightarrow \text{Int} \sim \forall X.X \rightarrow \text{Int} \\
\varepsilon_{\text{Int}} &= \langle \text{Int}, \text{Int} \rangle \Vdash \cdot; \cdot \quad \vdash \text{Int} \sim \text{Int} \\
\varepsilon_? &= \langle ?, ? \rangle \Vdash \cdot; X \quad \vdash ? \sim ? \\
\varepsilon_X &= \langle X, X \rangle \Vdash \cdot; X \quad \vdash X \sim ? \\
\varepsilon_{out} &= \langle \alpha \rightarrow \text{Int}, \text{Int} \rightarrow \text{Int} \rangle \Vdash \alpha := \text{Int}; \cdot \quad \vdash \alpha \rightarrow \text{Int} \sim \text{Int} \rightarrow \text{Int} \\
\varepsilon_{\alpha \rightarrow \alpha} &= \langle \alpha \rightarrow \alpha, \alpha \rightarrow \alpha \rangle \Vdash \alpha := \text{Int}; \cdot \quad \vdash \alpha \rightarrow \alpha \sim \alpha \rightarrow \alpha \\
\varepsilon_\alpha &= \langle \alpha, \alpha \rangle \Vdash \alpha := \text{Int}; \cdot \quad \vdash \alpha \sim ? \\
\langle \text{Int}, \alpha \rangle &\Vdash \alpha := \text{Int}; \cdot \quad \vdash \text{Int} \sim \alpha \\
\langle \text{Int}, ? \rangle &\Vdash \alpha := \text{Int}; \cdot \quad \vdash \text{Int} \sim ?
\end{aligned}$$

Initially, the first ascription to variable x , namely $\varepsilon_X x :: ?$, is deemed well-typed thanks to the following consistent judgment:

$$\varepsilon_X = \langle X, X \rangle \Vdash \cdot; X \vdash X \sim ?$$

Then, after type application, $\varepsilon_{out} = \langle \alpha \rightarrow \text{Int}, \text{Int} \rightarrow \text{Int} \rangle$ justifies that $(X \rightarrow \text{Int})[\alpha/X] = \alpha \rightarrow \text{Int}$ is consistent with $(X \rightarrow \text{Int})[\text{Int}/X] = \text{Int} \rightarrow \text{Int}$, *i.e.*

$$\langle \alpha \rightarrow \text{Int}, \text{Int} \rightarrow \text{Int} \rangle \Vdash \alpha := \text{Int}; \cdot \vdash \alpha \rightarrow \text{Int} \sim \text{Int} \rightarrow \text{Int}$$

Upon application, the argument $\varepsilon_{\text{Int}} 1 :: \text{Int}$ is ascribed to the expected type of the function α by combining ε_{Int} ($\varepsilon_{\text{Int}} \Vdash \cdot; \cdot \vdash \text{Int} \sim ?$) with the domain information of ε_{out} ($\text{dom}(\varepsilon_{out}) \Vdash \cdot; \cdot \vdash \text{Int} \sim \alpha$). Using the definition of consistent transitivity (Def. 3.15), $\langle \text{Int}, \text{Int} \rangle \wp \langle \text{Int}, \alpha \rangle = \langle \text{Int}, \alpha \rangle \Vdash \alpha := \text{Int}; \cdot \vdash \text{Int} \sim \alpha$.⁷ This operation corresponds to a sealing of the value. The sealed value is then substituted for x inside the body of the function.

For justifying that the value bound to x can be ascribed to $?$, we need evidence for $\text{Int} \sim ?$ by composing the two judgments below using consistent transitivity:

$$\langle \text{Int}, \alpha \rangle \Vdash \alpha := \text{Int}; \cdot \vdash \text{Int} \sim \alpha \quad \langle \alpha, \alpha \rangle \Vdash \alpha := \text{Int}; \cdot \vdash \alpha \sim ?$$

Note that the second judgment is obtained by substituting α for X in ε_X .

Using the definition of consistent transitivity (Def. 3.15), $\langle \text{Int}, \alpha \rangle \wp \langle \alpha, \alpha \rangle = \langle \text{Int}, \alpha \rangle$. Similarly, for justifying the second ascription to $?$, $\langle \text{Int}, \alpha \rangle$ must be combined with the evidence of the judgment for the second ascription:

$$\langle ?, ? \rangle \Vdash \alpha := \text{Int}; \cdot \vdash ? \sim ?$$

By Def. 3.15, $\langle \text{Int}, \alpha \rangle \wp \langle ?, ? \rangle = A^2(\{\langle \text{Int}, \text{Int} \rangle, \langle \text{Int}, \alpha \rangle\}) = \langle \text{Int}, ? \rangle$. This evidence can subsequently be used to produce evidence to justify that the addition is well-typed, since $\langle \text{Int}, ? \rangle \wp \langle \text{Int}, \text{Int} \rangle = \langle \text{Int}, \text{Int} \rangle$. Therefore the program produces 2, without errors: parametricity is violated.

3.5.2 Refining Evidence

For gradual parametricity, evidence must do more than just ensure type safety. It needs to safeguard the sealing that type variables are meant to represent, also taking care of unsealing as necessary. First of all, we need to define evidence to adequately represent consistency judgments of GSF.

⁷ Following Def. 3.15: Let $\Xi = \alpha := \text{Int}$, $\langle \text{Int}, \text{Int} \rangle \wp \langle \text{Int}, \alpha \rangle = A^2(\{\langle T_1, T_2 \rangle \in C(\text{Int}) \times C(\alpha) \mid \exists T_2 \in C(\text{Int}) \cap C(\alpha) \wedge \Sigma \in C(\Xi) \wedge \Sigma; \Delta \vdash T_1 = T_2 \wedge \Sigma; \Delta \vdash T_2 = T_3\})$, but $C(\text{Int}) = \{\text{Int}\}$, $C(\alpha) = \{\alpha\}$, and $C(\Xi) = \alpha := \text{Int}$. Then $\langle \text{Int}, \text{Int} \rangle \wp \langle \text{Int}, \alpha \rangle = A^2(\{\langle \text{Int}, \alpha \rangle \mid \Sigma \in C(\Xi) \wedge \Sigma; \Delta \vdash \text{Int} = \text{Int} \wedge \Sigma; \Delta \vdash \text{Int} = \alpha\}) = A^2(\{\langle \text{Int}, \alpha \rangle\}) = \langle \text{Int}, \alpha \rangle$.

$$\begin{aligned}
\text{lift}_{\Xi}(G) &= \begin{cases} B & G = B \\ X & G = X \\ ? & G = ? \\ \text{lift}_{\Xi}(G_1) \rightarrow \text{lift}_{\Xi}(G_2) & G = G_1 \rightarrow G_2 \\ \forall X. \text{lift}_{\Xi}(G_1) & G = \forall X. G_1 \\ \text{lift}_{\Xi}(G_1) \times \text{lift}_{\Xi}(G_2) & G = G_1 \times G_2 \\ \alpha^{\text{lift}_{\Xi}(\Xi(\alpha))} & G = \alpha \end{cases} \\
\text{unlift}(E) &= \begin{cases} B & E = B \\ X & E = X \\ ? & E = ? \\ \text{unlift}(E_1) \rightarrow \text{unlift}(E_2) & E = E_1 \rightarrow E_2 \\ \forall X. \text{unlift}(E_1) & E = \forall X. E_1 \\ \text{unlift}(E_1) \times \text{unlift}(E_2) & E = E_1 \times E_2 \\ \alpha & E = \alpha^{E_1} \end{cases}
\end{aligned}$$

Figure 3.7: Lifting Operations

Evidence Types. Instead of using gradual types in the representation of evidence, we introduce *evidence types* $E \in \text{ETYPE}$ with the following syntax:

$$E ::= B \mid E \rightarrow E \mid \forall X. E \mid E \times E \mid \alpha^E \mid X \mid ?$$

Then, we define an evidence ε as a pair of evidence types $\langle E_1, E_2 \rangle$. The only difference between gradual types and evidence types is in type names (highlighted in gray above). SF equality judgments, and hence GSF consistency judgments, are relative to a store. It is therefore not enough to use type names in evidence: we need to keep track of their associated types in the store. An evidence type name α^E captures the type associated to the type name α through the store. For instance, evidence that a variable has a polymorphic type X is initially $\langle X, X \rangle$. When X is instantiated, say to Int , and a fresh type name α is introduced, the evidence becomes $\langle \alpha^{\text{Int}}, \alpha^{\text{Int}} \rangle$. An evidence type name does not only record the end type to which it is bound, but the whole path. For instance, $\alpha^{\beta^{\text{Int}}}$ is a valid evidence type name that embeds the fact that α is bound to β , which is itself bound to Int .

Note that as a program reduces, evidence can not only become more precise than statically-used types, but also than the global store. For instance, it can be the case that $\alpha := ?$ in the global store Ξ , but that locally, the evidence for α has gotten more precise, such as α^{Int} . We use the definition $\text{lift}_{\Xi}(G)$ to enrich a type G with the type information in Ξ (Figure 3.7), returning an evidence type E . For instance, a type name is enriched recursively with the type that is instantiated in the store, $\text{lift}_{\Xi}(\alpha) = \alpha^{\text{lift}_{\Xi}(\Xi(\alpha))}$. Dually, unlifting ($\text{unlift}(E)$) forgets the additional information related to type instantiations, receiving an evidence type E and returning a gradual type G . For example, $\text{unlift}(\alpha^E) = \alpha$. In all other cases, both operations recur structurally (Figure 3.7).

It is crucial to understand the intuition behind the *position* of type names in a given evidence. The position of α^E in an evidence can correspond to a *sealing*, an *unsealing*, or

$$\begin{aligned}
s[\alpha^E/X] &= \begin{cases} b & s = b \\ \lambda x : G_1[\alpha/X].t[\alpha^E/X] & s = \lambda x : G_1.t \\ \Lambda Y.t[\alpha^E/X] & s = \Lambda Y.t \\ \langle s_1[\alpha^E/X], s_2[\alpha^E/X] \rangle & s = \langle s_1, s_2 \rangle \\ x & s = x \\ \varepsilon[\alpha^E/X]t[\alpha^E/X] :: G[\alpha/X] & s = \varepsilon t :: G \\ op(\overline{t[\alpha^E/X]}) & s = op(\bar{t}) \\ t_1[\alpha^E/X] t_2[\alpha^E/X] & s = t_1 t_2 \\ \pi_i(t[\alpha^E/X]) & s = \pi_i(t) \\ t[\alpha^E/X] [G[\alpha/X]] & s = t [G] \end{cases} \\
\varepsilon[\alpha^E/X] &= \langle \pi_1(\varepsilon)[\alpha^E/X], \pi_2(\varepsilon)[\alpha^E/X] \rangle \\
\pi_1(\langle E_1, E_2 \rangle) &= E_1 \quad \pi_2(\langle E_1, E_2 \rangle) = E_2 \\
E[\alpha^{E'}/X] &= \begin{cases} B & E = B \\ E_1[\alpha^{E'}/X] \rightarrow E_2[\alpha^{E'}/X] & E = E_1 \rightarrow E_2 \\ \forall Y.E_1[\alpha^{E'}/X] & E = \forall Y.E_1 \\ E_1[\alpha^{E'}/X] \times E_2[\alpha^{E'}/X] & E = E_1 \times E_2 \\ \alpha^{E_1[\alpha^{E'}/X]} & E = \alpha^{E_1} \\ \alpha^{E'} & E = X \\ Y & E = Y \wedge X \neq Y \\ ? & E = ? \end{cases}
\end{aligned}$$

Figure 3.8: Term and Evidence Type Substitution

neither. If α^E is *only* on the right side, *e.g.* $\langle \text{Int}, \alpha^{\text{Int}} \rangle$, then the evidence is a sealing (here, of Int with α). Dually, if α^E is *only* on the left side, *e.g.* $\langle \alpha^{\text{Int}}, \text{Int} \rangle$, the evidence is an unsealing (here, of Int from α). Sealing and unsealing evidences arise through reduction, as will be illustrated later in this section.

Armed with the precise definition of evidence, Figure 3.8 defines the term, evidence and evidence type substitution operations, used in the runtime semantics (Figure 3.5). Type substitution over a term is defined inductively over its sub-terms, and by consequence over their evidences. Observe that type substitution on the annotated types of a term must transform evidence types into gradual types by using the unlift operator ($\text{unlift}(\alpha^E) = \alpha$). This occurs in the type substitution on function, ascription, and type application terms. Type substitution in evidence is defined as the type substitution in each of its components, and evidence type substitution is defined inductively in the expected way.

Figure 3.9 defines evidence inversion functions. For instance, if ε justifies that $G_{11} \rightarrow G_{12} \sim G_{21} \rightarrow G_{22}$, then $\text{dom}(\varepsilon)$ computes new evidence that justifies that $G_{21} \sim G_{11}$, and $\text{cod}(\varepsilon)$ computes new evidence that justifies that $G_{12} \sim G_{22}$. Similarly, if ε justifies that $G_{11} \times G_{12} \sim G_{21} \times G_{22}$, then $p_i(\varepsilon)$ justifies that $G_{1i} \sim G_{2i}$. Finally, if ε_i justifies that $G_{1i} \sim G_{2i}$, then $\varepsilon_1 \times \varepsilon_2$ justifies that $G_{11} \times G_{12} \sim G_{21} \times G_{22}$.

$$\begin{aligned}
\text{dom}(\langle E_{11} \rightarrow E_{12}, E_{21} \rightarrow E_{22} \rangle) &= \langle E_{21}, E_{11} \rangle & \text{cod}(\langle E_{11} \rightarrow E_{12}, E_{21} \rightarrow E_{22} \rangle) &= \langle E_{12}, E_{22} \rangle \\
\text{dom}(\varepsilon) &\text{ undefined o/w} & \text{cod}(\varepsilon) &\text{ undefined o/w} \\
p_i(\langle E_{11} \times E_{12}, E_{21} \times E_{22} \rangle) &= \langle E_{1i}, E_{2i} \rangle & \langle E_{11}, E_{21} \rangle \times \langle E_{12}, E_{22} \rangle &= \langle E_{11} \times E_{12}, E_{21} \times E_{22} \rangle \\
p_i(\varepsilon) &\text{ undefined o/w} & &
\end{aligned}$$

Figure 3.9: Auxiliary Functions for Evidence

$\varepsilon \circledast \varepsilon = \varepsilon$ **Consistent transitivity**

$$\begin{aligned}
& \text{(base)} \frac{}{\langle B, B \rangle \circledast \langle B, B \rangle = \langle B, B \rangle} & \text{(typeVar)} \frac{}{\langle X, X \rangle \circledast \langle X, X \rangle = \langle X, X \rangle} \\
& \text{(idL)} \frac{}{\langle E_1, E_2 \rangle \circledast \langle ?, ? \rangle = \langle E_1, E_2 \rangle} & \text{(idR)} \frac{}{\langle ?, ? \rangle \circledast \langle E_1, E_2 \rangle = \langle E_1, E_2 \rangle} \\
& \text{(sealL)} \frac{\langle E_1, E_2 \rangle \circledast \langle E_3, E_4 \rangle = \langle E'_1, E'_2 \rangle}{\langle E_1, E_2 \rangle \circledast \langle E_3, \alpha^{E_4} \rangle = \langle E'_1, \alpha^{E'_2} \rangle} & \text{(sealR)} \frac{\langle E_1, E_2 \rangle \circledast \langle E_3, E_4 \rangle = \langle E'_1, E'_2 \rangle}{\langle \alpha^{E_1}, E_2 \rangle \circledast \langle E_3, E_4 \rangle = \langle \alpha^{E'_1}, E'_2 \rangle} \\
& \text{(unsl)} \frac{\langle E_1, E_2 \rangle \circledast \langle E_3, E_4 \rangle = \langle E'_1, E'_2 \rangle}{\langle E_1, \alpha^{E_2} \rangle \circledast \langle \alpha^{E_3}, E_4 \rangle = \langle E'_1, E'_2 \rangle} \\
& \text{(func)} \frac{\langle E_{41}, E_{31} \rangle \circledast \langle E_{21}, E_{11} \rangle = \langle E_3, E_1 \rangle \quad \langle E_{12}, E_{22} \rangle \circledast \langle E_{32}, E_{42} \rangle = \langle E_2, E_4 \rangle}{\langle E_{11} \rightarrow E_{12}, E_{21} \rightarrow E_{22} \rangle \circledast \langle E_{31} \rightarrow E_{32}, E_{41} \rightarrow E_{42} \rangle = \langle E_1 \rightarrow E_2, E_3 \rightarrow E_4 \rangle} \\
& \text{(abst)} \frac{\langle E_1, E_2 \rangle \circledast \langle E_3, E_4 \rangle = \langle E'_1, E'_2 \rangle}{\langle \forall X. E_1, \forall X. E_2 \rangle \circledast \langle \forall X. E_3, \forall X. E_4 \rangle = \langle \forall X. E'_1, \forall X. E'_2 \rangle} \\
& \text{(pair)} \frac{\langle E_{11}, E_{21} \rangle \circledast \langle E_{31}, E_{41} \rangle = \langle E_1, E_3 \rangle \quad \langle E_{12}, E_{22} \rangle \circledast \langle E_{32}, E_{42} \rangle = \langle E_2, E_4 \rangle}{\langle E_{11} \times E_{12}, E_{21} \times E_{22} \rangle \circledast \langle E_{31} \times E_{32}, E_{41} \times E_{42} \rangle = \langle E_1 \times E_2, E_3 \times E_4 \rangle}
\end{aligned}$$

$\varepsilon \sqsubseteq \varepsilon$ **Evidence precision**

$$\frac{E_1 \sqsubseteq E_3 \quad E_2 \sqsubseteq E_4}{\langle E_1, E_2 \rangle \sqsubseteq \langle E_3, E_4 \rangle}$$

$E \sqsubseteq E$ **Evidence type precision**

$$\begin{array}{c}
\frac{}{B \sqsubseteq B} \quad \frac{}{X \sqsubseteq X} \quad \frac{E_1 \sqsubseteq E_2}{\alpha^{E_1} \sqsubseteq \alpha^{E_2}} \quad \frac{}{B \sqsubseteq ?} \quad \frac{}{? \sqsubseteq ?} \quad \frac{E_1 \rightarrow E_2 \sqsubseteq ? \rightarrow ?}{E_1 \rightarrow E_2 \sqsubseteq ?} \\
\frac{E_1 \sqsubseteq E_3 \quad E_2 \sqsubseteq E_4}{E_1 \rightarrow E_2 \sqsubseteq E_3 \rightarrow E_4} \quad \frac{E_1 \times E_2 \sqsubseteq ? \times ?}{E_1 \times E_2 \sqsubseteq ?} \quad \frac{E_1 \sqsubseteq E_3 \quad E_2 \sqsubseteq E_4}{E_1 \times E_2 \sqsubseteq E_3 \times E_4} \\
\frac{E_1 \sqsubseteq E_2}{\forall X. E_1 \sqsubseteq \forall X. E_2} \quad \frac{\forall X. E \sqsubseteq \forall X. ?}{\forall X. E \sqsubseteq ?} \quad \frac{}{X \sqsubseteq ?} \quad \frac{}{\alpha^E \sqsubseteq ?}
\end{array}$$

Figure 3.10: Consistent Transitivity and Evidence Precision

Consistent Transitivity. With the syntactic enrichment of evidence types, consistent transitivity can be strengthened to account for sealing and unsealing, ensuring parametricity.

Consistent transitivity is defined inductively in Figure 3.10. Save for rules (idL) and (idR), these inductive rules are equivalent to the formal definition of consistent transitivity given by AGT (Def. 3.15). We describe the interesting rules next.

Rule (unsl) specifies that when a sealing and an unsealing of the same type name meet in the middle positions of a consistent transitivity step, the type name can be eliminated in order to calculate the resulting evidence. For instance, $\langle \text{Int}, \alpha^{\text{Int}} \rangle \mathbin{\text{;}} \langle \alpha^?, ? \rangle = \langle \text{Int}, \text{Int} \rangle \mathbin{\text{;}} \langle ?, ? \rangle = \langle \text{Int}, \text{Int} \rangle$.

As shown in §3.5.1, it is important for consistent transitivity to not lose precision when combining an evidence with an unknown evidence. To this end, and contrary to the formal definition given by AGT shown at the end of §3.5.1, rule (idL) in Fig. 3.10 preserves the left evidence. Going back to the example of §3.5.1, we now have $\langle \text{Int}, \alpha^{\text{Int}} \rangle \mathbin{\text{;}} \langle ?, ? \rangle = \langle \text{Int}, \alpha^{\text{Int}} \rangle$, instead of $\langle \text{Int}, ? \rangle$. Because $\langle \text{Int}, \alpha^{\text{Int}} \rangle \mathbin{\text{;}} \langle \text{Int}, \text{Int} \rangle$ is undefined, reduction steps to **error** as desired.

Rule (sealL) shows that when an evidence is combined with a sealing, the resulting evidence is also a sealing. This sealing can be more precise, *e.g.* $\langle \text{Int}, \text{Int} \rangle \mathbin{\text{;}} \langle ?, \alpha^? \rangle = \langle \text{Int}, \alpha^{\text{Int}} \rangle$.

There is one rule per type constructor. For example, rule (func) corresponds to the function case, where consistent transitivity is computed recursively with the domain and codomain evidences. Also, there are symmetric variants for some rules—such as (idR) and (sealR)—in which the left and right components of each evidence are swapped.

Evidence precision. Precision for evidence and evidence types is defined in Figure 3.10. The definition of evidence type precision is defined analogous to the definition of type precision, accounting as well for evidence type names α^E . We say that a type name is more precise than another if their bounded types are related by precision.

Properties. Importantly, this refined definition of consistent transitivity preserves associativity and optimality. It does however break monotonicity, and consequently, the dynamic gradual guarantee (we come back to this in §3.6).

Instantiation and Outer Evidence. The reduction rule of a type application ($R_{\text{app}}G$) produces two evidences. The first one is the instantiation evidence $\varepsilon[\hat{\alpha}]$ that justifies that the type of $t[\hat{\alpha}/X]$ ($G''[\alpha/X]$) is consistent with $G[\alpha/X]$. The second one is the outer evidence ε_{out} that justifies that $G[\alpha/X]$ is consistent with $G[G'/X]$:

$$\begin{aligned} \Xi \triangleright (\varepsilon \wedge X.t :: \forall X.G) [G'] &\longrightarrow \Xi' \triangleright \varepsilon_{\text{out}}(\varepsilon[\hat{\alpha}]t[\hat{\alpha}/X] :: G[\alpha/X]) :: G[G'/X] \text{ where} \\ \Xi' \triangleq \Xi, \alpha &:= G' \text{ for some } \alpha \notin \text{dom}(\Xi) \text{ and } \hat{\alpha} = \text{lift}_{\Xi'}(\alpha) \end{aligned}$$

Evidence $\varepsilon[\hat{\alpha}]$ is defined as: $\langle E_1, E_2 \rangle[\alpha^E] = \langle E_1[\alpha^E], E_2[\alpha^E] \rangle$, where the evidence types $E_i[\alpha^E]$ are obtained by the type application of E_i to α^E . The precise definition of ε_{out} is more delicate, addressing a subtle tension between ensuring the precision required for justifying unsealing when possible and not introducing new runtime errors when combined.

$$\varepsilon_{\text{out}} \triangleq \langle E_*[\alpha^E], E_*[E'] \rangle \quad \text{where } E_* = \text{lift}_{\Xi}(\text{unlift}(\pi_2(\varepsilon))), \alpha^E = \text{lift}_{\Xi'}(\alpha), E' = \text{lift}_{\Xi}(G')$$

In this definition, ε , α , G' , Ξ , and Ξ' come from rule (RappG). The evidence types $E_*[\alpha^E]$ and $E_*[E']$ are obtained by the type application of E_* to α^E and E' , respectively. Observe that E_* is obtained using the second component of evidence ε , and not using the information of the first component. This is because ε justifies that $\forall X.G''$ is consistent with $\forall X.G$, where G'' is the type of the body of the type abstraction, and $\forall X.G$ is the ascribed type. Therefore, evidence $\varepsilon[\hat{\alpha}]$ justifies that $G''[\alpha/X]$ is consistent with $G[\alpha/X]$, where the right (resp. left) component of $\varepsilon[\hat{\alpha}]$ corresponds to the most precise information about $G[\alpha/X]$ (resp. $G''[\alpha/X]$). As ε_{out} must justify that $G[\alpha/X]$ is consistent with $G[G'/X]$, we only use the information of the second component of ε (which corresponds to the most precise information about G). To illustrate this, consider $\varepsilon = \langle \forall X.X \rightarrow \text{Bool}, \forall X.X \rightarrow \beta^{\text{Bool}} \rangle$ and $\varepsilon[\hat{\alpha}] = \langle \alpha^{\text{Int}} \rightarrow \text{Bool}, \alpha^{\text{Int}} \rightarrow \beta^{\text{Bool}} \rangle$, then $\varepsilon_{out} = \langle \alpha^{\text{Int}} \rightarrow \beta^{\text{Bool}}, \text{Int} \rightarrow \beta^{\text{Bool}} \rangle$, and $\varepsilon[\hat{\alpha}] \circledast \varepsilon_{out} = \varepsilon_{out}$. If ε_{out} would have been constructed using the first component, then $\varepsilon_{out} = \langle \alpha^{\text{Int}} \rightarrow \text{Bool}, \text{Int} \rightarrow \beta^{\text{Bool}} \rangle$, but $\varepsilon[\hat{\alpha}] \circledast \varepsilon_{out}$ is not defined.

Determining E_* is the key challenge. The roundtrip unlift/lift “resets” the information of evidence type names to that contained in the store. As previously mentioned, the local information of evidence can be more precise than in the global store. Therefore, it is not necessarily true that $\text{lift}_{\Xi}(\text{unlift}(\pi_2(\varepsilon))) = \pi_2(\varepsilon)$. For example, if we take $\varepsilon = \langle \text{Int}, \alpha^{\text{Int}} \rangle$ and $\Xi(\alpha) = ?$, then $\pi_2(\varepsilon) = \alpha^{\text{Int}}$, $\text{unlift}(\pi_2(\varepsilon)) = \alpha$ and $\text{lift}_{\Xi}(\text{unlift}(\pi_2(\varepsilon))) = \alpha^?$. This technicality is crucial for proving parametricity (specifically, the compositionality lemma—see §3.7). It is important to note that although the first component of evidence ε_{out} can lose precision with respect the second component of ε , this local loss does not affect the precision of the entire program since evidence $\varepsilon[\hat{\alpha}]$ maintains the precision achieved so far. Furthermore, $\varepsilon[\hat{\alpha}] \circledast \varepsilon_{out}$ never fails: the role of the outer evidence ε_{out} is just to seal arguments supplied to the function, and unseal values returned by the function, and not to introduce new runtime errors; the possible runtime errors must come from the inner evidence $\varepsilon[\hat{\alpha}]$. For instance, evidence $\varepsilon_{out} = \langle \alpha^{\text{Int}} \rightarrow \alpha^{\text{Int}}, \text{Int} \rightarrow \text{Int} \rangle$ seals arguments when they are combined with $\text{dom}(\langle \alpha^{\text{Int}} \rightarrow \alpha^{\text{Int}}, \text{Int} \rightarrow \text{Int} \rangle) = \langle \text{Int}, \alpha^{\text{Int}} \rangle$, and unseals returned values when combined with $\text{cod}(\langle \alpha^{\text{Int}} \rightarrow \alpha^{\text{Int}}, \text{Int} \rightarrow \text{Int} \rangle) = \langle \alpha^{\text{Int}}, \text{Int} \rangle$. Note that ε_{out} will never cause a runtime error when combined with the resulting evidence of the parametric term result because both are necessarily related by precision. Indeed, by (RappG):

$$\Xi \triangleright (\varepsilon \lambda X.t :: \forall X.G) [G'] \longrightarrow \Xi' \triangleright \varepsilon_{out}(\varepsilon[\hat{\alpha}]t[\hat{\alpha}/X] :: G[\alpha/X]) :: G[G'/X]$$

and by (Rf):

$$\frac{\Xi' \triangleright \varepsilon[\hat{\alpha}]t[\hat{\alpha}/X] :: G[\alpha/X] \mapsto^* \Xi'' \triangleright \varepsilon'u :: G[\alpha/X]}{\Xi' \triangleright \varepsilon_{out}(\varepsilon[\hat{\alpha}]t[\hat{\alpha}/X] :: G[\alpha/X]) :: G[G'/X] \mapsto^* \Xi' \triangleright \varepsilon_{out}(\varepsilon'u :: G[\alpha/X]) :: G[G'/X]}$$

Since $\pi_2(\varepsilon') \sqsubseteq \pi_2(\varepsilon[\hat{\alpha}]) \sqsubseteq \pi_1(\varepsilon_{out})$, the combination $(\varepsilon' \circledast \varepsilon_{out})$ through transitivity never fails.

Illustration. The following reduction trace illustrates all the important aspects of reduction. Recall that we use ε_G to denote the evidence that justifies the reflexive judgment $G \sim G$. For example, $\varepsilon_{\alpha \rightarrow \alpha}$ justifies the reflexive judgment $\alpha \rightarrow \alpha \sim \alpha \rightarrow \alpha$ under environment $\Xi = \alpha := \text{Int}$.

$$\begin{aligned} \varepsilon_{\alpha \rightarrow \alpha} \Vdash \Xi; \cdot \vdash \alpha \rightarrow \alpha \sim \alpha \rightarrow \alpha \\ \text{where } \varepsilon_{\alpha \rightarrow \alpha} = \langle \text{lift}_{\Xi}(\alpha \rightarrow \alpha), \text{lift}_{\Xi}(\alpha \rightarrow \alpha) \rangle = \langle \alpha^{\text{Int}} \rightarrow \alpha^{\text{Int}}, \alpha^{\text{Int}} \rightarrow \alpha^{\text{Int}} \rangle \end{aligned}$$

$$\begin{array}{l}
(\varepsilon_{\forall X.X \rightarrow X}(\Lambda X.\lambda x : X.x) :: \forall X.X \rightarrow ?) \text{ [Int]} (\varepsilon_{\text{Int}} 1 :: \text{Int}) \\
\text{initial evidence} \\
(\text{Rapp}G) \mapsto \langle \langle \alpha^{\text{Int}} \rightarrow \alpha^{\text{Int}}, \text{Int} \rightarrow \text{Int} \rangle (\varepsilon_{\alpha \rightarrow \alpha}(\lambda x : \alpha.x) :: \alpha \rightarrow ?) :: \text{Int} \rightarrow ? \rangle (\varepsilon_{\text{Int}} 1 :: \text{Int}) \\
\varepsilon_{out} \text{ and } \varepsilon[\hat{\alpha}] \text{ are computed} \\
(\text{Rasc}) \mapsto \langle \langle \alpha^{\text{Int}} \rightarrow \alpha^{\text{Int}}, \text{Int} \rightarrow \text{Int} \rangle (\lambda x : \alpha.x) :: \text{Int} \rightarrow ? \rangle (\varepsilon_{\text{Int}} 1 :: \text{Int}) \\
\text{consistent transitivity} \\
(\text{Rapp}) \mapsto \langle \alpha^{\text{Int}}, \text{Int} \rangle (\langle \text{Int}, \alpha^{\text{Int}} \rangle 1 :: \alpha) :: ? \\
\text{argument is sealed} \\
(\text{Rasc}) \mapsto \langle \text{Int}, \text{Int} \rangle 1 :: ? \\
\text{unsealing eliminates } \alpha
\end{array}$$

Crucially, the initial evidence of the identity function is fully precise, even though it is ascribed an imprecise type. Consequently, in the first reduction step above, ε_{out} is calculated as:

$$\varepsilon_{out} \triangleq \langle E_*[\alpha^E], E_*[E'] \rangle = \langle (\forall X.X \rightarrow X)[\alpha^{\text{Int}}], (\forall X.X \rightarrow X)[\text{Int}] \rangle = \langle \alpha^{\text{Int}} \rightarrow \alpha^{\text{Int}}, \text{Int} \rightarrow \text{Int} \rangle$$

The application step (*Rapp*) then gives rise to sealing and unsealing evidences after deconstructing ε_{out} : the inner evidence $\langle \text{Int}, \alpha^{\text{Int}} \rangle$ seals the number 1 at type α , while the outer evidence $\langle \alpha^{\text{Int}}, \text{Int} \rangle$ allows the subsequent unsealing in the ascription step (*Rasc*). As a result, the ascribed identity function yields usable values, because the outer evidence subsequently takes care of unsealing. This addresses the violation of the dynamic gradual guarantee reported with λB and System F_C in §2.4. Note that if the function explicitly introduced imprecision, *e.g.* $\Lambda X.\lambda x : X.(x :: ?)$, then initial evidence would likewise be imprecise, and deconstructing ε_{out} would *not* justify unsealing the result anymore.

The following reduction trace illustrates all the important aspects of reduction.

$$\begin{array}{ll}
(\varepsilon_{\forall X.X \rightarrow ?}(\Lambda X.\lambda x : X.(\varepsilon_X x :: ?)) :: \forall X.X \rightarrow ?) \text{ [Int]} (\varepsilon_{\text{Int}} 1 :: \text{Int}) & \text{initial evidence} \\
(\text{Rapp}G) \mapsto \langle \langle \alpha^{\text{Int}} \rightarrow ?, \text{Int} \rightarrow ? \rangle (\varepsilon_{\alpha \rightarrow ?}(\lambda x : \alpha.(\varepsilon_\alpha x :: ?)) :: \alpha \rightarrow ?) :: \text{Int} \rightarrow ? \rangle (\varepsilon_{\text{Int}} 1 :: \text{Int}) & \text{type application} \\
(\text{Rasc}) \mapsto \langle \langle \alpha^{\text{Int}} \rightarrow ?, \text{Int} \rightarrow ? \rangle (\lambda x : \alpha.(\varepsilon_\alpha x :: ?)) :: \text{Int} \rightarrow ? \rangle (\varepsilon_{\text{Int}} 1 :: \text{Int}) & \text{consistent transitivity} \\
(\text{Rapp}) \mapsto \varepsilon_?(\varepsilon_\alpha(\langle \text{Int}, \alpha^{\text{Int}} \rangle 1 :: \alpha) :: ?) :: ? & \text{argument is sealed} \\
(\text{Rasc}) \mapsto \varepsilon_?(\langle \text{Int}, \alpha^{\text{Int}} \rangle 1 :: ?) :: ? & \text{consistent transitivity} \\
(\text{Rasc}) \mapsto \langle \text{Int}, \alpha^{\text{Int}} \rangle 1 :: ? & \text{unsealing does not occur}
\end{array}$$

We will return to a similar example in Section 3.6, which studies the dynamic gradual guarantee and why GSF does not fully satisfy it.

3.5.3 Basic Properties of GSF Evaluation

The runtime semantics of a GSF term are given by first translating the term to $\text{GSF}\varepsilon$ (noted $\vdash t \rightsquigarrow t_\varepsilon : G$) and then reducing the $\text{GSF}\varepsilon$ term. We write $t \Downarrow \Xi \triangleright v$ (resp. $t \Downarrow \mathbf{error}$) if

$\vdash t \rightsquigarrow t_\varepsilon : G$ and $\cdot \triangleright t_\varepsilon \mapsto^* \Xi \triangleright v$ (resp. $\cdot \triangleright t_\varepsilon \mapsto^* \mathbf{error}$) for some resulting store Ξ . We write $\Xi \triangleright v : G$ for $\Xi; \cdot \vdash v : G$. We write $t \uparrow$ if the translation of t diverges, and $t \Downarrow v$ when the store is irrelevant.

The properties of GSF follow from the same properties of GSF_ε , expressed using the small-step reduction relation, due to the fact that the translation \rightsquigarrow preserves typing. In particular, GSF terms do not get stuck, although they might produce **error** or diverge:

Proposition 3.16 (Type Safety) *If $\vdash t : G$ then either $t \Downarrow \Xi \triangleright v$ with $\Xi \triangleright v : G$, $t \Downarrow \mathbf{error}$, or $t \uparrow$.*

Proposition 3.10 established that GSF typing coincides with SF typing on static terms. A similar result holds considering the dynamic semantics. In particular, static GSF terms never produce **error**:

Proposition 3.17 (Static terms do not fail) *Let t be a static term. If $\vdash t : T$ then $\neg(t \Downarrow \mathbf{error})$.*

This result follows from the fact that all evidences in a static program are static, hence never gain precision; the initial type checking ensures that combination through transitivity never fails. This result can be found in the Appendix.

3.6 GSF and the Dynamic Gradual Guarantee

The previous section clarified several aspects of the semantics of GSF programs, by establishing type safety, and by showing that static terms do not fail. This section studies the dynamic gradual guarantee (DGG) (Siek et al., 2015a), also known as graduality (New and Ahmed, 2018; New et al., 2020). In a big-step setting, this guarantee essentially says that if $\vdash t : G$ and $t \Downarrow v$, then for any t' such that $t \sqsubseteq t'$, we have $t' \Downarrow v'$ for some v' such that $v \sqsubseteq v'$. Intuitively: losing precision is harmless, or, reducibility is monotonic with respect to precision.

Unfortunately, in order to enforce parametricity (§3.7), and as already alluded to earlier (§3.5.2), GSF does not satisfy the DGG. First, we exhibit a counterexample, and identify the non-monotonicity of consistent transitivity as the root cause for this behavior. Then, in order to better understand the behavior of GSF programs when losing precision, we study a weaker variant of the DGG—weaker in the sense that it is valid for a stricter notion of precision—first in GSF_ε (§3.6.3) and then in GSF. The idea of devising a stricter notion of precision for which a variant of the DGG can be satisfied was first explored by Igarashi et al. (2017a), even though they leave the proof of such a result for System F_G as a conjecture; here, we formally prove that GSF satisfies the DGG with respect to the strict notion of precision.

3.6.1 Violation of the Dynamic Gradual Guarantee in GSF

To show that GSF does not satisfy the dynamic gradual guarantee (DGG), it is sufficient to exhibit two terms in GSF, related by precision, whose behavior contradicts the DGG. Consider the polymorphic identity function $id_X \triangleq \Lambda X. \lambda x : X. x :: X$, and an imprecise variant $id_? \triangleq \Lambda X. \lambda x : ?.x :: X$. Then $id_X \text{ [Int]} 1 \Downarrow 1$, but $id_? \text{ [Int]} 1 \Downarrow \mathbf{error}$, despite the fact that $id_X \text{ [Int]} 1 \sqsubseteq id_? \text{ [Int]} 1$.

Conceptually, it is interesting to shed light on what causes such a violation. Recall that Garcia et al. (2016) prove the DGG for their language using (mostly) the monotonicity of consistent transitivity (Prop 3.14 (c)) with respect to imprecision. In fact, while not sufficient, we can prove that monotonicity of *consistent transitivity* (CT) is a *necessary* condition for the DGG to hold. Intuitively, since two successive ascriptions are collapsed via consistent transitivity, a violation of monotonicity for consistent transitivity immediately implies a violation of monotonicity for reduction, and hence a violation of the DGG. For instance, if $t = \varepsilon_2(\varepsilon_1 u :: G_1) :: G_2$, $t' = \varepsilon'_2(\varepsilon'_1 u' :: G'_1) :: G'_2$, $t \sqsubseteq t'$ and $t \mapsto (\varepsilon_1 \mathbin{\&} \varepsilon_2)u :: G_2$, then by the DGG, $t' \mapsto (\varepsilon'_1 \mathbin{\&} \varepsilon'_2)u :: G'_2$, and thus $\varepsilon'_1 \mathbin{\&} \varepsilon'_2$ should be defined.

Proposition 3.18 (\neg monotonicity of CT $\Rightarrow \neg$ DGG) *Let $\varepsilon_1 \sqsubseteq \varepsilon'_1$, $\varepsilon_2 \sqsubseteq \varepsilon'_2$, $\varepsilon_1 \Vdash G_1 \sim G_2$, $\varepsilon_2 \Vdash G_2 \sim G_3$, $\varepsilon'_1 \Vdash G'_1 \sim G'_2$, $\varepsilon'_2 \Vdash G'_2 \sim G'_3$, where $G_i \sqsubseteq G'_i$. If $\varepsilon_1 \mathbin{\&} \varepsilon_2 \not\sqsubseteq \varepsilon'_1 \mathbin{\&} \varepsilon'_2$, then $\exists t \sqsubseteq t'$, such that $t \mapsto v$, $t' \mapsto v'$ such that $v \not\sqsubseteq v'$.*

PROOF. Let $t \triangleq \varepsilon_2(\varepsilon_1 u :: G_2) :: G_3$, and $t' \triangleq \varepsilon'_2(\varepsilon'_1 u' :: G'_2) :: G'_3$, for some $u \sqsubseteq u'$. We know $t \sqsubseteq t'$. Let $\varepsilon_1 \mathbin{\&} \varepsilon_2 = \varepsilon_{12}$ and $\varepsilon'_1 \mathbin{\&} \varepsilon'_2 = \varepsilon'_{12}$, then $t \mapsto \varepsilon_{12}u :: G_3$ and $t' \mapsto \varepsilon'_{12}u' :: G'_3$, but as $\varepsilon_{12} \not\sqsubseteq \varepsilon'_{12}$ then $\varepsilon_{12}u :: G_3 \not\sqsubseteq \varepsilon'_{12}u' :: G'_3$ and the result holds. \square

Garcia et al. (2016) study a language without universal types. But in GSF, because of universal types, there is an additional monotonicity condition that is necessary for the DGG to hold: monotonicity of *evidence instantiation* (EI). Monotonicity of EI states that given two type abstractions related by precision, the new evidences created after type application remain related. Intuitively, since type application uses evidence instantiation, a violation of monotonicity for the latter implies a violation of monotonicity for the former, and hence a violation of the DGG. Formally:

Proposition 3.19 (\neg monotonicity of EI $\Rightarrow \neg$ DGG) *Let $\varepsilon_1 \sqsubseteq \varepsilon_2$, $G_1 \sqsubseteq G_2$, $\Xi_1 \sqsubseteq \Xi_2$, $\alpha := G_1 \in \Xi_1$, $\alpha := G_2 \in \Xi_2$, $\hat{\alpha}_1 = \text{lift}_{\Xi_1}(\alpha)$, $\hat{\alpha}_2 = \text{lift}_{\Xi_2}(\alpha)$, and $\varepsilon_1[\hat{\alpha}_1]$ is defined. If $\varepsilon_1[\hat{\alpha}_1] \not\sqsubseteq \varepsilon_2[\hat{\alpha}_2]$, or $\varepsilon_{1out} \not\sqsubseteq \varepsilon_{2out}$, then $\exists t \sqsubseteq t'$, such that $t \mapsto v$, $t' \mapsto v'$ such that $v \not\sqsubseteq v'$.*

PROOF. Let $t \triangleq (\varepsilon_1(\Lambda X. t_1) :: \forall X. G'_1) [G_1]$, and $t' \triangleq (\varepsilon_2(\Lambda X. t_2) :: \forall X. G'_2) [G_2]$, for some $t_1 \sqsubseteq t_2$ and $\forall X. G'_1 \sqsubseteq \forall X. G'_2$. We know $t \sqsubseteq t'$. Also, we know that $\Xi_1 \triangleright t \mapsto \Xi_1, \alpha := G_1 \triangleright \varepsilon_{1out}(\varepsilon_1[\hat{\alpha}_1]t'_1 :: G'_1[\alpha/X]) : G'_1[G_1/X]$ and $\Xi_2 \triangleright t' \mapsto \Xi_2, \alpha := G_2 \triangleright \varepsilon_{2out}(\varepsilon_2[\hat{\alpha}_2]t'_2 :: G'_2[\alpha/X]) : G'_2[G_2/X]$, but as either $\varepsilon_{1out} \not\sqsubseteq \varepsilon_{2out}$ or $\varepsilon_1[\hat{\alpha}_1] \not\sqsubseteq \varepsilon_2[\hat{\alpha}_2]$, then $\varepsilon_{1out}(\varepsilon_1[\hat{\alpha}_1]t'_1 :: G'_1[\alpha/X]) : G'_1[G_1/X] \not\sqsubseteq \varepsilon_{2out}(\varepsilon_2[\hat{\alpha}_2]t'_2 :: G'_2[\alpha/X]) : G'_2[G_2/X]$, and the result holds. \square

As mentioned in §3.5.2, monotonicity of consistent transitivity is broken by the strengthening we impose to enforce parametricity. For instance, consider $\langle \text{Int}, \alpha^{\text{Int}} \rangle \sqsubseteq \langle \text{Int}, \alpha^{\text{Int}} \rangle$ and

$\langle \alpha^{\text{Int}}, \text{Int} \rangle \sqsubseteq \langle ?, ? \rangle$. By consistent transitivity, $\langle \text{Int}, \alpha^{\text{Int}} \rangle \circ \langle \alpha^{\text{Int}}, \text{Int} \rangle = \langle \text{Int}, \text{Int} \rangle$ (rule unsl), and $\langle \text{Int}, \alpha^{\text{Int}} \rangle \circ \langle ?, ? \rangle = \langle \text{Int}, \alpha^{\text{Int}} \rangle$ (rule idL), but $\langle \text{Int}, \text{Int} \rangle \not\sqsubseteq \langle \text{Int}, \alpha^{\text{Int}} \rangle$. Therefore the DGG cannot be satisfied as such. We later on discuss a tension between our notion of parametricity and the DGG (§3.7), but first, we look at how to characterize the set of terms for which loss of precision is indeed harmless in GSF.

3.6.2 Towards a Weak Dynamic Gradual Guarantee for GSF

One way to accommodate the dynamic gradual guarantee in languages like λB , GSF, and System F_G , would be to change the definition of type (and term) precision. This is the approach taken by Igarashi et al. (2017a), although they do not prove that the DGG holds with this adjusted precision, and leave it as a conjecture. Dually, if one sticks to the natural notion of precision, as adopted by both GSF and CSA, and justified by the AGT interpretation. Reconciliation might come from considering other forms of parametricity, or perhaps less flexible gradual language designs (Devriese et al., 2018) (we will explore this other path in Part II). Here, inspired by the approach of Igarashi et al. (2017a), we devise an alternative notion of precision for which the DGG does hold. We call this relation *strict precision* as it relates fewer terms than the natural notion of precision. Conversely to Igarashi et al. (2017a), however, we do not intend strict precision to be the one used to typecheck programs, but only to serve as a technical device to characterize harmless losses of precision in GSF.

External vs internal losses of precision. First of all, it is important to observe that the violation of the DGG from the previous section is due to the interaction between polymorphic types and imprecision, which affects runtime sealing with type names. A first consequence of this observation is that the simply-typed subset of GSF should enjoy the DGG with respect to the standard notion of precision. Said differently, strict precision ought to coincide with natural precision on simply-typed terms. A second consequence is that excluding *any* loss of precision related to type variables would be a sound approximation to characterize when the DGG holds; this corresponds exactly to the precision relation of System F_G (Igarashi et al., 2017a). However, while this approach would work for GSF as well, it appears too strict, in that it excludes losses of precision on polymorphic types which are harmless in GSF.

Intuitively, we observe that in GSF, losing precision *internally* (*i.e.* by modifying the types of binders) has a different impact on reducibility, compared to losing precision *externally* (*i.e.* through imprecise type ascriptions). Specifically, external loss of precision is harmless in GSF when the ascribed term is closed with respect to type variables. Therefore any fully-static polymorphic function that is imprecisely ascribed and used adequately (type-wise) in a gradual context will behave as expected. In practice, this means that in GSF, the fully precise polymorphic identity function $id_X \triangleq \Lambda X. \lambda x : X. x :: X$ and an imprecisely-ascribed variant such as $id_{X?} \triangleq id_X :: \forall X. ? \rightarrow X$ have the same behavior—in particular, one can apply $id_{X?}$ to any given type and argument of that type and successfully obtain back that argument as result. In contrast, as we have seen, the internally-imprecise function $id_? \triangleq \Lambda X. \lambda x : ?.x :: X$ fails when applied, because the argument value is not sealed on entry, and hence the unsealing on exit is invalid.

Admittedly, this difference in behavior between internal and external losses of precision

might come as a surprise to programmers, but it is the result of type-driven sealing. When applying one of the functions above, the operational semantics must decide whether or not the value bound to x ought to be sealed. If the type of x is *known* to be X , as in id_X and $id_{X?}$, it is clear that the value should be sealed (with the runtime type name corresponding to X). However, for $id_?$, the type of x is $?$, so there are two options: not sealing because $?$ might stand for other types than X , or sealing because $?$ might stand for X . Always sealing presents two issues. First, if the function was $\Lambda X.\lambda x : ?.x + 1$, then the addition would fail at runtime, and we would have another counterexample of the DGG (because it is less precise than $\Lambda X.\lambda x : \text{Int}.x + 1$). More importantly, we could not know with respect to *which variable* one ought to seal. Indeed, consider a slightly more complex function: $\Lambda X.\Lambda Y.\lambda z : ?.t$. Here, always sealing would require deciding whether to seal with (the runtime names of) X or Y .

This conundrum arises because runtime sealing is type driven, and types can be imprecise. When faced with an imprecise term binder under a type binder, either options of sealing or not sealing would expose a failure of the DGG. Optimistically not sealing has the advantage of avoiding the ambiguity of which type names to seal with, while still supporting harmless losses of precision externally for polymorphic values. Note that a language design such as PolyG $^\nu$ in which sealing and unsealing are independent of the precision of type information can sidestep the problem, by leaving the task of sealing with explicit terms to programmers (§2.4).

Characterizing strict term precision for GSF. Strict term precision should coincide with natural precision on simply-typed terms, but how should it behave on the polymorphic fragment of GSF? We now provide some intuitive characterization of strict term precision, denoted \leq , based on the analysis above with the three terms id_X , $id_{X?}$, and $id_?$.

(A) $id_X \not\leq id_?$

$id_?$ presents an internal loss of precision compared to id_X , because the term binder changes from type X to type $?$, and fails at runtime when applied.

(B) $id_X :: \forall X.X \rightarrow X \leq id_{X?}$

$id_{X?}$ presents an external loss of precision compared to $id_X :: \forall X.X \rightarrow X$, and does not fail when applied.

(C) $id_X :: \forall X.X \rightarrow X \leq id_? :: \forall X.X \rightarrow X$

Although $id_?$ presents an internal loss of precision compared to id_X , the ascription to $\forall X.X \rightarrow X$ on $id_?$ imposes this parametricity contract, and hence the resulting term does behave like a proper (static) identity function.

The rest of this section builds upon this informal analysis in order to fully define strict precision and establish the corresponding dynamic gradual guarantee. Of course, because strict precision \leq is more restrictive than standard precision \sqsubseteq , the dynamic gradual guarantee that one may establish with respect to it is *weaker*; hereafter, we denote it DGG^{\leq} . The dynamic gradual guarantee appeals to term reduction, so in §3.6.3 we start by defining strict precision for GSF_ε , prove DGG^{\leq} for GSF_ε , and conclude by establishing DGG^{\leq} for GSF. Finally, in §3.6.4 we provide a characterization of \leq directly on GSF syntax, *i.e.* without appealing to elaboration, for which DGG^{\leq} holds.

$G \leq G$ **Strict type precision**

$$\begin{array}{c}
\frac{}{B \leq B} \quad \frac{}{X \leq X} \quad \frac{}{\alpha \leq \alpha} \quad \frac{}{B \leq ?} \quad \frac{G_1 \rightarrow G_2 \leq ? \rightarrow ?}{G_1 \rightarrow G_2 \leq ?} \quad \frac{}{? \leq ?} \\
\frac{G_1 \leq G_3 \quad G_2 \leq G_4}{G_1 \rightarrow G_2 \leq G_3 \rightarrow G_4} \quad \frac{G_1 \leq G_3 \quad G_2 \leq G_4}{G_1 \times G_2 \leq G_3 \times G_4} \quad \frac{G_1 \leq G_2}{\forall X. G_1 \leq \forall X. G_2}
\end{array}$$

$\varepsilon \leq \varepsilon$ **Strict evidence precision**

$$\frac{E_1 \leq E_3 \quad E_2 \leq E_4}{\langle E_1, E_2 \rangle \leq \langle E_3, E_4 \rangle}$$

$E \leq E$ **Strict evidence type precision**

$$\begin{array}{c}
\frac{}{B \leq B} \quad \frac{}{X \leq X} \quad \frac{E_1 \leq E_2}{\alpha^{E_1} \leq \alpha^{E_2}} \quad \frac{}{B \leq ?} \quad \frac{E_1 \rightarrow E_2 \leq ? \rightarrow ?}{E_1 \rightarrow E_2 \leq ?} \\
\frac{}{? \leq ?} \quad \frac{E_1 \leq E_3 \quad E_2 \leq E_4}{E_1 \rightarrow E_2 \leq E_3 \rightarrow E_4} \quad \frac{E_1 \leq E_3 \quad E_2 \leq E_4}{E_1 \times E_2 \leq E_3 \times E_4} \quad \frac{E_1 \leq E_2}{\forall X. E_1 \leq \forall X. E_2}
\end{array}$$

Figure 3.11: GSF: Strict Precision

3.6.3 Weak Dynamic Gradual Guarantee for GSF

Armed with the intuition presented above, we define a strict notion of precision for GSF_ε , which closely characterizes GSF_ε terms for which monotonicity of consistent transitivity holds. While not sufficient, monotonicity of consistent transitivity is necessary for the DGG to hold, as established in Prop. 3.18.

Strict precision for gradual types. Strict precision for types (Figure 3.11) avoids any interference between runtime sealing and loss of precision. As expected, \leq coincides with \sqsubseteq except for universal types, type variables and type names: these are not more precise than the unknown type anymore. For instance, $\forall X. X \rightarrow X \not\leq \forall X. X \rightarrow ? \not\leq ?$. We say G_1 is “more strictly precise” than G_2 when $G_1 \leq G_2$.

Strict precision for evidence. Strict type precision can be naturally lifted to define strict precision for evidence and evidence types (Figure 3.11). A type name is more strictly precise than another if it is bound to a more strictly precise evidence type. Crucially, monotonicity of consistent transitivity holds with respect to \leq .

Proposition 3.20 (\leq -Monotonicity of Consistent Transitivity) *If $\varepsilon_1 \leq \varepsilon_2$, $\varepsilon_3 \leq \varepsilon_4$, and $\varepsilon_1 \circ \varepsilon_3$ is defined, then $\varepsilon_1 \circ \varepsilon_3 \leq \varepsilon_2 \circ \varepsilon_4$.*

For illustration purposes, let us recall the counterexample to monotonicity presented in §3.6.1. Consider $\langle \text{Int}, \alpha^{\text{Int}} \rangle \sqsubseteq \langle \text{Int}, \alpha^{\text{Int}} \rangle$ and $\langle \alpha^{\text{Int}}, \text{Int} \rangle \sqsubseteq \langle ?, ? \rangle$. By consistent transitivity, $\langle \text{Int}, \alpha^{\text{Int}} \rangle \circ \langle \alpha^{\text{Int}}, \text{Int} \rangle = \langle \text{Int}, \text{Int} \rangle$ (rule unsl), and $\langle \text{Int}, \alpha^{\text{Int}} \rangle \circ \langle ?, ? \rangle = \langle \text{Int}, \alpha^{\text{Int}} \rangle$ (rule idL), but $\langle \text{Int}, \text{Int} \rangle \not\sqsubseteq \langle \text{Int}, \alpha^{\text{Int}} \rangle$. This argument is no longer valid with strict precision, as $\alpha^{\text{Int}} \not\leq ?$ and

therefore $\langle \alpha^{\text{Int}}, \text{Int} \rangle \not\leq \langle ?, ? \rangle$.

$\boxed{\Omega \vdash \Xi \triangleright s : G \leq \Xi \triangleright s : G}$ **Strict term precision** (for conciseness, s ranges over both t and u)

$$\begin{array}{c}
(\leq b_\varepsilon) \frac{ty(b) = B \quad \Xi_1 \leq \Xi_2}{\Omega \vdash \Xi_1 \triangleright b : B \leq \Xi_2 \triangleright b : B} \\
(\leq \lambda_\varepsilon) \frac{\Omega, x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright t_1 : G'_1 \leq \Xi_2 \triangleright t_2 : G'_2 \quad G_1 \sqsubseteq G_2}{\Omega \vdash \Xi_1 \triangleright \lambda x : G_1.t_1 : G_1 \rightarrow G'_1 \leq \Xi_2 \triangleright \lambda x : G_2.t_2 : G_2 \rightarrow G'_2} \\
(\leq \times_\varepsilon) \frac{\Omega \vdash \Xi_1 \triangleright s_1 : G_1 \leq \Xi_2 \triangleright s_2 : G_2 \quad \Omega \vdash \Xi_1 \triangleright s'_1 : G'_1 \leq \Xi_2 \triangleright s'_2 : G'_2}{\Omega \vdash \Xi_1 \triangleright \langle s_1, s'_1 \rangle : G_1 \times G'_1 \leq \Xi_2 \triangleright \langle s_2, s'_2 \rangle : G_2 \times G'_2} \\
(\leq \Lambda_\varepsilon) \frac{\Omega \vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2}{\Omega \vdash \Xi_1 \triangleright \Lambda X.t_1 : \forall X.G_1 \leq \Xi_2 \triangleright \Lambda X.t_2 : \forall X.G_2} \quad (\leq x_\varepsilon) \frac{x : G_1 \sqsubseteq G_2 \in \Omega \quad \Xi_1 \leq \Xi_2}{\Omega \vdash \Xi_1 \triangleright x : G_1 \leq \Xi_2 \triangleright x : G_2} \\
(\leq \text{op}_\varepsilon) \frac{\Omega \vdash \Xi_1 \triangleright t_1 : \overline{G} \leq \Xi_2 \triangleright t_2 : \overline{G} \quad ty(\text{op}) = \overline{G} \rightarrow G'}{\Omega \vdash \Xi_1 \triangleright \text{op}(\overline{t_1}) : G' \leq \Xi_2 \triangleright \text{op}(\overline{t_2}) : G'} \\
(\leq \text{app}_\varepsilon) \frac{\Omega \vdash \Xi_1 \triangleright t_1 : G'_1 \rightarrow G_1 \leq \Xi_2 \triangleright t_2 : G'_2 \rightarrow G_2 \quad \Omega \vdash \Xi_1 \triangleright t'_1 : G'_1 \leq \Xi_2 \triangleright t'_2 : G'_2}{\Omega \vdash \Xi_1 \triangleright t_1 t'_1 : G_1 \leq \Xi_2 \triangleright t_2 t'_2 : G_2} \\
(\leq \text{appG}_\varepsilon) \frac{\Omega \vdash \Xi_1 \triangleright t_1 : \forall X.G_1 \leq \Xi_2 \triangleright t_2 : \forall X.G_2 \quad G'_1 \leq G'_2}{\Omega \vdash \Xi_1 \triangleright t_1 [G'_1] : G_1[G'_1/X] \leq \Xi_2 \triangleright t_2 [G'_2] : G_2[G'_2/X]} \\
(\leq \text{pair}_\varepsilon) \frac{\Omega \vdash \Xi_1 \triangleright t_1 : G_1 \times G_2 \leq \Xi_2 \triangleright t_2 : G'_1 \times G'_2}{\Omega \vdash \Xi_1 \triangleright \pi_i(t_1) : G_i \leq \Xi_2 \triangleright \pi_i(t_2) : G'_i} \\
(\leq \text{asc}_\varepsilon) \frac{\varepsilon_1 \leq \varepsilon_2 \quad \Omega \vdash \Xi_1 \triangleright s_1 : G'_1 \leq \Xi_2 \triangleright s_2 : G'_2 \quad G_1 \sqsubseteq G_2}{\Omega \vdash \Xi_1 \triangleright \varepsilon_1 s_1 :: G_1 : G_1 \leq \Xi_2 \triangleright \varepsilon_2 s_2 :: G_2 : G_2} \\
(\leq \text{Masc}_\varepsilon) \frac{\varepsilon_1 \sqsubseteq \varepsilon_2 \quad \Omega \vdash \Xi_1 \triangleright t_1 : G'_1 \leq \Xi_2 \triangleright t_2 : G'_2 \quad G_1 \sqsubseteq G_2 \quad \varepsilon_1 = \mathcal{J}_{\Xi_1}(G_1, G_1) \quad \varepsilon_2 = \mathcal{J}_{\Xi_2}(G_2, G_2) \quad G'_1 \rightarrow G_1 \quad G'_2 \rightarrow G_2}{\Omega \vdash \Xi_1 \triangleright \varepsilon_1 t_1 :: G_1 : G_1 \leq \Xi_2 \triangleright \varepsilon_2 t_2 :: G_2 : G_2}
\end{array}$$

$\boxed{\Xi \vdash t \leq \Xi \vdash t}$ **Configuration precision**

$$\begin{array}{c}
\frac{\Xi_1 \leq \Xi_2 \quad \cdot \vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2 \quad \Xi_1 \vdash t_1 : G_1 \quad \Xi_2 \vdash t_2 : G_2}{\Xi_1 \triangleright t_1 \leq \Xi_2 \triangleright t_2} \\
\frac{\forall \alpha \in \text{dom}(\Xi_1). \Xi_1(\alpha) \leq \Xi_2(\alpha)}{\Xi_1 \leq \Xi_2}
\end{array}$$

Figure 3.12: GSF ε : Strict Precision

Strict precision for GSF ε terms. Strict precision for GSF ε terms relates two possibly-open terms and their respective types (Figure 3.12). It is worth noting that types can

be related by \leq or \sqsubseteq depending on the rule. We use \sqsubseteq to relate every type annotation (save for type instantiations), and \leq for (almost) every other relation. Note that we could have obtained a simpler definition by using \leq everywhere, but the relation would be overly conservative, for instance rejecting example (C) among many others. Our goal is to design a term precision relation as permissive as possible (close to the natural term precision relation) such that the DGG is satisfied. The precision judgment $\Omega \vdash \Xi_1 \triangleright s_1 : G_1 \leq \Xi_2 \triangleright s_2 : G_2$ denotes that term s_1 of type G_1 is more strictly precise than s_2 of type G_2 , under store Ξ_1 strictly more precise than Ξ_2 , and precision relation environment Ω . It is important to clarify that the judgment $\Omega \vdash \Xi_1 \triangleright s_1 : G_1 \leq \Xi_2 \triangleright s_2 : G_2$ does not necessarily imply that $G_1 \leq G_2$. In fact, the types G_1 and G_2 are related by the more general relation \sqsubseteq . Nevertheless, evidences are the ones that play a crucial role in the term precision relation, which are related in almost every case by \leq . Ω binds a term variable to a pair of types related by precision \sqsubseteq . The intuition about why we use \sqsubseteq and not \leq in Ω is that as long as evidence are related by \leq , we can relax this relation on type annotations. Furthermore, using \leq on Ω makes the precision relation on terms overly conservative rejecting example (C) above. Rule ($\leq x_\varepsilon$) establishes $\Omega \vdash \Xi_1 \triangleright x : G_1 \leq \Xi_2 \triangleright x : G_2$ if $x : G_1 \sqsubseteq G_2 \in \Omega$, and Rule ($\leq \lambda_\varepsilon$) extends Ω with the annotated types of the functions to relate.

Strict term precision is the natural lifting of strict type precision \leq to terms, except for types that do not influence evidence in the runtime semantics, namely function argument types and ascription types: for these, we can use the more liberal type precision relation \sqsubseteq . Note that these types are used to elaborate evidence, but at runtime once evidence is elaborated, they are no longer relevant, unlike instantiation types which propagate to evidence upon application. For example, Rule ($\leq \text{asc}_\varepsilon$) has the premise $G_1 \sqsubseteq G_2$. If we imposed a strict precision relation between ascribed types, then example (B) would not be satisfied as $\forall X.X \rightarrow X \not\leq \forall X.? \rightarrow X$. By ($\leq \text{asc}_\varepsilon$) we know that for the elaborated terms $\varepsilon_{\forall X.X \rightarrow X} \text{id}_X :: \forall X.X \rightarrow X \leq \varepsilon_{\forall X.X \rightarrow X} \text{id}_X :: \forall X.? \rightarrow X$ because evidences are the same (and thus related by \leq), whereas the type annotations are related by \sqsubseteq . Furthermore, example (A) is satisfied as the elaborated terms are not related by strict precision because $\varepsilon_{\forall X.X \rightarrow X}$ is not related to $\varepsilon_{\forall X.? \rightarrow X}$.

Rule ($\leq \text{appG}_\varepsilon$) states that types involved in a type application must be related by strict precision because they do influence evidence during reduction: after elimination of type abstractions, new evidences are created using these types, and such evidences need to be related as well. Note that this restriction is sufficient to satisfy monotonicity of evidence instantiation, which is needed for the dynamic gradual guarantee (Prop 3.19).

Finally, we need to strengthen the relation with an additional rule ($\leq \text{Masc}_\varepsilon$) to account for GSF_ε terms that are the result of the elaboration from GSF. This will be important to scale the DGG^\leq from GSF_ε to GSF below. Recall that the translation from GSF to GSF_ε introduces evidences to ensure that GSF_ε terms are well-typed (Figure 3.6). In particular, the translation uses type matching \rightarrow to ascribe subterms of type $?$ in elimination positions to the corresponding top-level type constructor. When an actual matching expansion occurs, the corresponding evidence is generated such as $\varepsilon_{\forall X.?} = \mathcal{G}_\Xi(\forall X.?, \forall X.?)$, or $\varepsilon_{? \rightarrow ?} = \mathcal{G}_\Xi(? \rightarrow ?, ? \rightarrow ?)$. Such evidences are related by \sqsubseteq , but not necessarily by \leq . Rule ($\leq \text{Masc}_\varepsilon$) accounts for the case where they are not. Note that evidences ε_1 and ε_2 do not contribute to any increase in precision: when combined with some arbitrary evidence ε during reduction, the combination $\varepsilon \ ; \ \varepsilon_i$ either fails or results in ε . Rule ($\leq \text{Masc}_\varepsilon$) is key to satisfy example

(B). To see why, consider terms $(id_X :: \forall X.X \rightarrow X)$ [Int] and $(id_X :: \forall X.? \rightarrow X)$ [Int] and their elaborations:

$$\begin{array}{c}
\text{(GappG)} \frac{\text{(Gascu)} \frac{id_X \rightsquigarrow id_{X'} \quad \varepsilon_1 = \mathcal{G}(\forall X.X \rightarrow X, \forall X.X \rightarrow X)}{id_X :: \forall X.X \rightarrow X \rightsquigarrow \varepsilon_1 id_X :: \forall X.X \rightarrow X} \quad \varepsilon_2 = \mathcal{G}(\forall X.X \rightarrow X, \forall X.X \rightarrow X)}{(id_X :: \forall X.X \rightarrow X) \text{ [Int]} \rightsquigarrow \varepsilon_2(\varepsilon_1 id_{X'} :: \forall X.X \rightarrow X) :: \forall X.X \rightarrow X \text{ [Int]}} \\
\\
\text{(GappG)} \frac{\text{(Gascu)} \frac{id_X \rightsquigarrow id_{X'} \quad \varepsilon'_1 = \mathcal{G}(\forall X.X \rightarrow X, \forall X.? \rightarrow X)}{id_X :: \forall X.? \rightarrow X \rightsquigarrow \varepsilon'_1 id_X :: \forall X.? \rightarrow X} \quad \varepsilon'_2 = \mathcal{G}(\forall X.? \rightarrow X, \forall X.? \rightarrow X)}{(id_X :: \forall X.? \rightarrow X) \text{ [Int]} \rightsquigarrow \varepsilon'_2(\varepsilon'_1 id_{X'} :: \forall X.? \rightarrow X) :: \forall X.? \rightarrow X \text{ [Int]}}
\end{array}$$

Note that $\varepsilon_1 = \varepsilon_2 = \varepsilon'_1 = \varepsilon_{\forall X.X \rightarrow X} = \langle \forall X.X \rightarrow X, \forall X.X \rightarrow X \rangle$, $\varepsilon'_2 = \langle \forall X.? \rightarrow X, \forall X.? \rightarrow X \rangle$, and $\varepsilon_2 \sqsubseteq \varepsilon'_2$ but $\varepsilon_2 \not\leq \varepsilon'_2$. After one step of execution, both pairs of evidences are combined ($\varepsilon_1 \mathbin{\&} \varepsilon_2$ and $\varepsilon'_1 \mathbin{\&} \varepsilon'_2$), resulting in both cases in $\varepsilon_{\forall X.X \rightarrow X}$. Therefore, these two programs behave identically and are thus related.

Strict precision for configurations. Figure 3.12 also defines strict type precision for GSF_ε stores and configurations. A store is more strictly precise than another if it binds each type name to a more strictly precise type. Finally, a configuration is more strictly precise than another if the store and term components are more strictly precise, and the terms are well-typed with their respective stores.

DGG \leq for GSF ε . Armed with strict precision for GSF_ε , and the fact that consistent transitivity is monotone with respect to it (Proposition 3.20), we can prove the weak dynamic gradual guarantee $\text{DGG}\leq$ for GSF_ε . Given two configurations related by strict precision, small-step reduction (\mapsto) of the most precise one implies that of the less precise one. Alternatively, if the first configuration is already a value, then so is the second.

Proposition 3.21 (Small-step $\text{DGG}\leq$ for GSF_ε) *Suppose $\Xi_1 \triangleright t_1 \leq \Xi_2 \triangleright t_2$.*

- a. *If $\Xi_1 \triangleright t_1 \mapsto \Xi'_1 \triangleright t'_1$, then $\Xi_2 \triangleright t_2 \mapsto \Xi'_2 \triangleright t'_2$, for some Ξ'_2 and t'_2 such that $\Xi'_1 \triangleright t'_1 \leq \Xi'_2 \triangleright t'_2$.*
- b. *If $t_1 = v_1$, then $t_2 = v_2$.*

DGG \leq for GSF. Using Proposition 3.21 we can establish $\text{DGG}\leq$ for GSF, considering that two GSF terms are related by strict precision iff their elaboration to GSF_ε are.

Theorem 3.22 ($\text{DGG}\leq$) *Suppose $t_1 \leq t_2$, $\vdash t_1 : G_1$, and $\vdash t_2 : G_2$.*

- a. *If $t_1 \Downarrow \Xi_1 \triangleright v_1$, then $t_2 \Downarrow \Xi_2 \triangleright v_2$, $\cdot \vdash \Xi_1 \triangleright v_1 : G_1 \leq \Xi_2 \triangleright v_2 : G_2$ and $\Xi_1 \leq \Xi_2$, for some v_2 and Ξ_2 .
If $t_1 \Uparrow$ then $t_2 \Uparrow$.*
- b. *If $t_2 \Downarrow \Xi_2 \triangleright v_2$, then $t_1 \Downarrow \Xi_1 \triangleright v_1$, $\cdot \vdash \Xi_1 \triangleright v_1 : G_1 \leq \Xi_2 \triangleright v_2 : G_2$ and $\Xi_1 \leq \Xi_2$, for some v_1 and Ξ_1 , or $t_1 \Downarrow$ **error**.
If $t_2 \Uparrow$, then $t_1 \Uparrow$ or $t_1 \Downarrow$ **error**.*

Harmless imprecise ascriptions. Finally, we can use the DGG^{\leq} to establish that, given a term t of type G , ascribing to a less precise type G' and then back to type G , results in a term semantically equivalent to t :

Lemma 3.23 *Let $\vdash t : G$, $G \sqsubseteq G'$, and $t' = t :: G' :: G$, then*

- $t \Downarrow \Xi \triangleright v \iff t' \Downarrow \Xi \triangleright v$.
- $t \Downarrow \mathbf{error} \iff t' \Downarrow \mathbf{error}$.

Note in particular that if t produces a value, then t' produces the exact same value. The dynamic semantics of $\text{GSF}\varepsilon$ ensure that t and $t :: A$ have equivalent behavior; both reduce to the same value, diverge or raise an error. Therefore, by transitivity, t and $t :: A :: A$ also have equivalent behavior. With this result and the DGG^{\leq} , we can prove that t and $t :: B :: A$ have equivalent behaviors (Lemma 3.23).

While the above result characterizes an ascription roundtrip through imprecision and back, we can also establish harmlessness results for imprecise ascriptions. Given a term t that reduces to some value, ascribing it to a less precise type also results in a (less strictly precise) value.

Lemma 3.24 *Let $\vdash t : G$ such that $t \Downarrow \Xi \triangleright v$, and $G \sqsubseteq G'$. Then $t :: G' \Downarrow \Xi \triangleright v'$ such that $\vdash \Xi \triangleright v : G \leq \Xi \triangleright v' : G'$, for some v' .*

Likewise, we can characterize ascribing the subterms of elimination forms, such as function application and type application:

Lemma 3.25 *Let $\vdash t_1 : G_1$ and $\vdash t_2 : G_2$ such that $\vdash t_1 t_2 : G$ and $t_1 t_2 \Downarrow \Xi \triangleright v$. Let $G_1 \sqsubseteq G'_1$, $G_2 \sqsubseteq G'_2$, and $G \sqsubseteq G'$, such that $\vdash (t_1 :: G'_1) (t_2 :: G'_2) : G'$. Then $(t_1 :: G'_1) (t_2 :: G'_2) \Downarrow \Xi \triangleright v'$ such that $\vdash \Xi \triangleright v : G \leq \Xi \triangleright v' : G'$, for some v' .*

Lemma 3.26 *Let $\vdash t : G_1$ such that $\vdash t [G_2] : G$ and $t [G_2] \Downarrow \Xi \triangleright v$. Let $G_1 \sqsubseteq G'_1$, $G_2 \leq G'_2$, and $G \sqsubseteq G'$, such that $\vdash (t :: G'_1) [G'_2] : G'$. Then $(t :: G'_1) [G'_2] \Downarrow \Xi' \triangleright v'$ such that $\vdash \Xi \triangleright v : G \leq \Xi' \triangleright v' : G'$ and $\Xi \leq \Xi'$, for some v' and Ξ' .*

Similar lemmas can be defined for other eliminations forms, such as projections and n-ary operations.

These results, which embody the motto that external imprecision is harmless in GSF , constitute a valuable compositionality guarantee when embedding fully-static (System F) terms in a gradual world, as will be further illustrated in §3.7.4.

3.6.4 Syntactic Strict Precision for GSF

For now, strict precision for GSF terms has been defined by appealing to their elaboration to $\text{GSF}\varepsilon$ terms. Unfortunately, with this definition it would be hard for programmers to get an intuition about when two terms are related by strict precision, as it would require understanding the elaborations to $\text{GSF}\varepsilon$. Here we design a strict precision relation \leq for GSF terms syntactically, as a sound approximation of the elaboration-based definition. To define a syntactic strict precision relation for

$\Omega \vdash v : G \leq_v v : G$ **Syntactic strict value precision**

$$\begin{array}{c}
(\leq b) \frac{ty(b) = B}{\Omega \vdash b : B \leq_v b : B} \quad (\leq \lambda) \frac{\Omega, x : G_1 \sqsubseteq G_2 \vdash t_1 : G'_1 \leq t_2 : G'_2 \quad G_1 \sqsubseteq G_2}{\Omega \vdash (\lambda x : G_1.t_1) : G_1 \rightarrow G'_1 \leq_v (\lambda x : G_2.t_2) : G_2 \rightarrow G'_2} \\
(\leq \lambda) \frac{\Omega, x : G_1 \sqsubseteq G_2 \vdash t_1 : G'_1 \leq t_2 : G'_2 \quad G_1 \sqsubseteq G_2}{\Omega \vdash (\lambda x : G_1.t_1) : G_1 \rightarrow G'_1 \leq_v (\lambda x : G_2.t_2) : G_2 \rightarrow G'_2} \\
(\leq \times) \frac{\Omega \vdash v_1 : G_1 \leq v_2 : G_2 \quad \Omega \vdash v'_1 : G'_1 \leq v'_2 : G'_2}{\Omega \vdash \langle v_1, v'_1 \rangle : G_1 \times G'_1 \leq_v \langle v_2, v'_2 \rangle : G_2 \times G'_2} \\
(\leq \Lambda) \frac{\Omega \vdash t_1 : G_1 \leq t_2 : G_2}{\Omega \vdash (\Lambda X.t_1) : \forall X.G_1 \leq_v (\Lambda X.t_2) : \forall X.G_2}
\end{array}$$

$\Omega \vdash t : G \leq t : G$ **Syntactic strict term precision**

$$\begin{array}{c}
(\leq x) \frac{x : G_1 \sqsubseteq G_2 \in \Omega}{\Omega \vdash x : G_1 \leq x : G_2} \quad (\leq v) \frac{\Omega \vdash v_1 : G_1 \leq_v v_2 : G_2 \quad G_1 \leq G_2}{\Omega \vdash v_1 : G_1 \leq v_2 : G_2} \\
(\leq \text{ascv}) \frac{\Omega \vdash v_1 : G'_1 \leq_v v_2 : G'_2 \quad G'_1 \sqcap G_1 \leq G'_2 \sqcap G_2 \quad G_1 \sqsubseteq G_2}{\Omega \vdash v_1 :: G_1 : G_1 \leq v_2 :: G_2 : G_2} \\
(\leq \text{asct}) \frac{\Omega \vdash t_1 : G'_1 \leq t_2 : G'_2 \quad G'_1 \sqcap G_1 \leq G'_2 \sqcap G_2 \quad G_1 \sqsubseteq G_2 \quad t_1, t_2 \neq v}{\Omega \vdash t_1 :: G_1 : G_1 \leq t_2 :: G_2 : G_2} \\
(\leq \text{op}) \frac{\Omega \vdash t_1 : \overline{G_1} \leq t_2 : \overline{G_2} \quad ty(op) = \overline{G} \rightarrow G' \quad \overline{G} \sqcap \overline{G_1} \leq \overline{G} \sqcap \overline{G_2}}{\Omega \vdash op(\overline{t_1}) : G'_1 \leq op(\overline{t_2}) : G'_2} \\
(\leq \text{app}) \frac{\Omega \vdash t_1 : G_1 \leq t_2 : G_2 \quad \Omega \vdash t'_1 : G'_1 \leq t'_2 : G'_2 \quad G_1 \rightarrow G_{11} \rightarrow G_{12} \quad G_2 \rightarrow G_{21} \rightarrow G_{22} \quad G'_1 \sqcap G_{11} \leq G'_2 \sqcap G_{21}}{\Omega \vdash t_1 t'_1 : G_{12} \leq t_2 t'_2 : G_{22}} \\
(\leq \text{pairt}) \frac{(t_1 \neq v_1 \vee t_2 \neq v_2) \quad \Omega \vdash t_1 : G_1 \leq t_2 : G_2 \quad \Omega \vdash t'_1 : G'_1 \leq t'_2 : G'_2}{\Omega \vdash \langle t_1, t'_1 \rangle : G_1 \times G'_1 \leq \langle t_2, t'_2 \rangle : G_2 \times G'_2} \\
(\leq \text{appG}) \frac{\Omega \vdash t_1 : G_1 \leq t_2 : G_2 \quad G_1 \rightarrow \forall X.G''_1 \quad G_2 \rightarrow \forall X.G''_2 \quad G'_1 \leq G'_2}{\Omega \vdash t_1 [G'_1] : G''_1[G'_1/X] \leq t_2 [G'_2] : G''_2[G'_2/X]} \\
(\leq \text{pairi}) \frac{\Omega \vdash t_1 : G_1 \leq t_2 : G_2 \quad G_1 \rightarrow G_{11} \times G_{21} \quad G_2 \rightarrow G_{12} \times G_{22}}{\Omega \vdash \pi_i(t_1) : G_{i1} \leq \pi_i(t_2) : G_{i2}}
\end{array}$$

$\Omega \vdash \Gamma \sqsubseteq \Gamma$ **Well-formedness Ω**

$$\frac{}{\cdot \vdash \cdot \sqsubseteq \cdot} \quad \frac{\Omega \vdash \Gamma_1 \sqsubseteq \Gamma_2 \quad G_1 \sqsubseteq G_2}{\Omega, x : G_1 \sqsubseteq G_2 \vdash \Gamma_1, x : G_1 \sqsubseteq \Gamma_2, x : G_2}$$

Figure 3.13: GSF: Syntactic Strict Term Precision

GSF, we start from the GSF to GSF ε translation rules, and analyze when two terms yield related elaborations. Let us first look at two crucial cases: ascriptions and type applications.

Ascriptions. For a couple of ascriptions $t_1 :: G'_1$ and $t_2 :: G'_2$ we know that:

$$\text{(Gasct)} \frac{\begin{array}{c} t_1 \neq v \\ \Delta; \Gamma_1 \vdash t_1 \rightsquigarrow t'_1 : G'_1 \quad \varepsilon_1 = \mathcal{I}(G'_1, G_1) \end{array}}{\Delta; \Gamma_1 \vdash t_1 :: G_1 \rightsquigarrow \varepsilon_1 t'_1 :: G_1 : G_1} \quad \text{(Gasct)} \frac{\begin{array}{c} t_2 \neq v \\ \Delta; \Gamma_2 \vdash t_2 \rightsquigarrow t'_2 : G'_2 \quad \varepsilon_2 = \mathcal{I}(G'_2, G_2) \end{array}}{\Delta; \Gamma_2 \vdash t_2 :: G_2 \rightsquigarrow \varepsilon_2 t'_2 :: G_2 : G_2}$$

If $\Omega \vdash \cdot \triangleright \varepsilon_1 t'_1 :: G_1 : G_1 \leq \cdot \triangleright \varepsilon_2 t'_2 :: G_2 : G_2$, then by ($\leq_{\text{asc}_\varepsilon}$), it must be the case that $\varepsilon_1 \leq \varepsilon_2$, $\Omega \vdash \cdot \triangleright t'_1 : G'_1 \leq \cdot \triangleright t'_2 : G'_2$ and $G_1 \sqsubseteq G_2$, where Ω is well-formed with respect to Γ_1 and Γ_2 , *i.e.* $\Omega \vdash \Gamma_1 \sqsubseteq \Gamma_2$ (Figure 3.13). As $\mathcal{I}(G'_1, G_1) = \langle G'_1 \sqcap G_1, G'_1 \sqcap G_1 \rangle$ and $\mathcal{I}(G'_2, G_2) = \langle G'_2 \sqcap G_2, G'_2 \sqcap G_2 \rangle$, then we require that $G'_1 \sqcap G_1 \leq G'_2 \sqcap G_2$, which leads to the following strict precision rule for ascriptions on GSF:

$$\frac{\Omega \vdash t_1 : G'_1 \leq t_2 : G'_2 \quad G'_1 \sqcap G_1 \leq G'_2 \sqcap G_2 \quad G_1 \sqsubseteq G_2 \quad t_1, t_2 \neq v}{\Omega \vdash t_1 :: G_1 : G_1 \leq t_2 :: G_2 : G_2}$$

Type applications. For a couple of type applications $t_1[G'_1]$ and $t_2[G'_2]$, we know from the elaboration rules that:

$$\text{(GappG)} \frac{\begin{array}{c} \Delta; \Gamma_1 \vdash t_1 \rightsquigarrow t'_1 : G_1 \quad \Delta \vdash G'_1 \\ G_1 \rightarrow \forall X. G''_1 \quad \varepsilon_1 = \mathcal{I}(G_1, \forall X. G''_1) \end{array}}{\Delta; \Gamma_1 \vdash t_1 [G'_1] \rightsquigarrow (\varepsilon_1 t'_1 :: \forall X. G''_1) [G'_1] : G''_1[G'_1/X]} \quad \text{(GappG)} \frac{\begin{array}{c} \Delta; \Gamma_2 \vdash t_2 \rightsquigarrow t'_2 : G_2 \quad \Delta \vdash G'_2 \\ G_2 \rightarrow \forall X. G''_2 \quad \varepsilon_2 = \mathcal{I}(G_2, \forall X. G''_2) \end{array}}{\Delta; \Gamma_2 \vdash t_2 [G'_2] \rightsquigarrow (\varepsilon_2 t'_2 :: \forall X. G''_2) [G'_2] : G''_2[G'_2/X]}$$

Let us suppose that $\Omega \vdash \cdot \triangleright (\varepsilon_1 t'_1 :: \forall X. G''_1) [G'_1] : G''_1[G'_1/X] \leq \cdot \triangleright (\varepsilon_2 t'_2 :: \forall X. G''_2) [G'_2] : G''_2[G'_2/X]$, where $\Omega \vdash \Gamma_1 \sqsubseteq \Gamma_2$. Then by ($\leq_{\text{appG}_\varepsilon}$), we know that $\Omega \vdash \cdot \triangleright (\varepsilon_1 t'_1 :: \forall X. G''_1) : \forall X. G''_1 \leq \cdot \triangleright (\varepsilon_2 t'_2 :: \forall X. G''_2) : \forall X. G''_2$ and $G'_1 \leq G'_2$. Note that $\mathcal{I}(G_i, \forall X. G''_i) = \mathcal{I}(\forall X. G''_i, \forall X. G''_i) = \langle \forall X. G''_i, \forall X. G''_i \rangle$, for $i \in \{1, 2\}$. By ($\leq_{\text{Masc}_\varepsilon}$), it must be the case that $\forall X. G''_1 \sqsubseteq \forall X. G''_2$ and $\Omega \vdash \cdot \triangleright t'_1 : G_1 \leq \cdot \triangleright t'_2 : G_2$. Finally, the strongest requirements yield the following strict precision rule for type applications:

$$\frac{\Omega \vdash t_1 : G_1 \leq t_2 : G_2 \quad G_1 \rightarrow \forall X. G''_1 \quad G_2 \rightarrow \forall X. G''_2 \quad G'_1 \leq G'_2}{\Omega \vdash t_1 [G'_1] : G''_1[G'_1/X] \leq t_2 [G'_2] : G''_2[G'_2/X]}$$

Syntactic strict precision. Figure 3.13 defines syntactic strict precision for GSF terms, which soundly reflects strict precision for GSF ε and can account for the translation of GSF terms to GSF ε . Judgment $\Omega \vdash t_1 : G_1 \leq t_2 : G_2$ denotes that term t_1 of type G_1 is more strictly precise to t_2 of type G_2 , under precision relation environment Ω . Note that contrary to GSF ε , we do not require type stores because source terms only exist prior to evaluation, and hence do not contain type names. Most of the rules are straightforward and derived following the reasoning explained above for

ascriptions and type applications. We use metavariable v in GSF to range over constants, functions and type abstractions, and use \leq_v to relate them. We make such distinction between precision on values and terms, because pair of values such as $\Lambda X.\lambda x : X.x :: X$ and $\Lambda X.\lambda x : ?.x :: X$ should not be related (\leq_v), but their ascriptions to $\forall X.X \rightarrow X$ should (\leq_{ascv}).

Rule (\leq_v) demands that the internal types of the values be related in \leq because we do not know in which context the value is going to be used. In contrast, rule (\leq_{ascv}) is more permissive, establishing that the internal types can be in \sqsubseteq —but only if the values have ascriptions such that their meet (*i.e.* initial evidences) are in \leq (as explained on how we derive (\leq_{asct})). This allows capturing some internal losses of precision, whenever the surrounding type information ensures that the associated evidence will be related by \leq . For instance, $(\Lambda X.\lambda x : X.x :: X) :: \forall X.X \rightarrow X \leq (\Lambda X.\lambda x : ?.x :: X) :: \forall X.X \rightarrow X$ at the corresponding type.

Rule (\leq_{asct}) uses the same technique to be as permissive as possible: it only requires $G_1 \sqsubseteq G_2$, but requires the meets of the types involved in the ascriptions to be related by \leq as explained before. Likewise, Rule (\leq_{app}) requires the meets of the function argument types and the actual argument types to be related by \leq . Note that during translation from GSF to $\text{GSF}\varepsilon$, the arguments t'_1 and t'_2 will be ascribed to $\text{dom}^\sharp(G_1)$ and $\text{dom}^\sharp(G_2)$ respectively. To account for strict precision over the evidence of the ascriptions $\mathcal{I}(G'_1, \text{dom}^\sharp(G_1)) = \langle G'_1 \sqcap \text{dom}^\sharp(G_1), G'_1 \sqcap \text{dom}^\sharp(G_1) \rangle$ and $\mathcal{I}(G'_2, \text{dom}^\sharp(G_2)) = \langle G'_2 \sqcap \text{dom}^\sharp(G_2), G'_2 \sqcap \text{dom}^\sharp(G_2) \rangle$, we require that $G'_1 \sqcap \text{dom}^\sharp(G_1) \leq G'_2 \sqcap \text{dom}^\sharp(G_2)$. Rule (\leq_{appG}) follows the $\text{GSF}\varepsilon$ precision rule for type instantiation and uses \leq to relate the instantiation types.

Soundness of syntactic strict precision. Finally, syntactic strict term precision for GSF is sound with respect to strict term precision of the translated terms in $\text{GSF}\varepsilon$:

Proposition 3.27 *Suppose t_1 and t_2 GSF terms such that $\cdot \vdash t_1 : G_1 \leq t_2 : G_2$, and their elaborations $\cdot \vdash t_1 \rightsquigarrow t_{\varepsilon_1} : G_1$ and $\cdot \vdash t_2 \rightsquigarrow t_{\varepsilon_2} : G_2$. Then $\cdot \vdash \triangleright t_{\varepsilon_1} : G_1 \leq \triangleright t_{\varepsilon_2} : G_2$.*

3.7 Gradual Parametricity for GSF

In this section, we first discuss two different notions of parametricity for gradual languages that have been developed in the literature (§3.7.1), in order to situate the notion of gradual parametricity for GSF (§3.7.2). Then, we show in §3.7.3 that this notion of gradual parametricity for GSF is incompatible with the DGG. This tension is solely driven by the definition of parametricity, and not by monotonicity of consistent transitivity (§3.6.1). This suggests that the incompatibility is shared by other languages with essentially the same notion of gradual parametricity, for which the dynamic gradual guarantee has so far been left as an open question. Finally, we explore gradual free theorems in GSF based on examples discussed in the literature, using both gradual parametricity and the DGG^\leq in order to establish such results (§3.7.4).

3.7.1 On Gradual Parametricities

We first review the standard technique to state and prove parametricity. The notion of parametricity established by Reynolds (1983) is usually defined by interpreting types as *binary logical relations*. The fundamental property of such a relation, also known as the *abstraction theorem*, states that

a well-typed term is related to itself at its type. Consequently, polymorphic terms must behave uniformly at all possible type instantiations.

Preliminaries. The definition of parametricity for the statically-typed polymorphic lambda calculus is standard and uncontroversial. Notationally, we follow Ahmed et al. (2017) in all the technical development hereafter. The chosen notations scale smoothly to describe gradual parametricity, both in other work and ours. The relational interpretation of types is presented using *atoms* of the form $(t_1, t_2) \in \text{Atom}[T_1, T_2]$, denoting that the closed terms t_1 and t_2 have types T_1 and T_2 , respectively. Formally:

$$\text{Atom}[T_1, T_2] = \{(t_1, t_2) \mid \cdot \vdash t_1 : T_1 \wedge \cdot \vdash t_2 : T_2\}$$

The logical relation is defined using two mutually-defined interpretations: one for values and one for computations. For simplicity and uniformity, throughout this section we use notation $(v_1, v_2) \in \mathcal{V}_\rho[[T]]$ when v_1 and v_2 are related values at type T under environment ρ , and notation $(t_1, t_2) \in \mathcal{J}_\rho[[T]]$ when terms t_1 and t_2 are related computations at type T under environment ρ . An environment ρ , which maps a type variable to two types and a relation, is used to relate values at abstract types as explained below. For convenience, we introduce the following notation for projections in ρ : if $\rho = \{X \mapsto (T_{11}, T_{12}, R_1), Y \mapsto (T_{21}, T_{22}, R_2), \dots\}$, then $\rho_1 = \{X \mapsto T_{11}, Y \mapsto T_{21}, \dots\}$, $\rho_2 = \{X \mapsto T_{12}, Y \mapsto T_{22}, \dots\}$, and $\rho_R = \{X \mapsto R_1, Y \mapsto R_2, \dots\}$.

Let us briefly go through the definitions. Two base values (of type B) are related if they are the same:

$$\mathcal{V}_\rho[[B]] = \{(v, v) \in \text{Atom}_\rho[B]\}$$

where $\text{Atom}_\rho[T] = \{(t_1, t_2) \mid (t_1, t_2) \in \text{Atom}[\rho_1(T), \rho_2(T)]\}$. Two functions are related if given two related argument the application yield related computations:

$$\mathcal{V}_\rho[[T_1 \rightarrow T_2]] = \{(v_1, v_2) \in \text{Atom}_\rho[T_1 \rightarrow T_2] \mid \forall (v'_1, v'_2) \in \mathcal{V}_\rho[[T_1]]. (v_1 \ v'_1, v_2 \ v'_2) \in \mathcal{J}_\rho[[T_2]]\}$$

Two type abstractions are related if their instantiations to two arbitrary types yield related computations for any given relation between the instantiated types:

$$\mathcal{V}_\rho[[\forall X. T]] = \{(v_1, v_2) \in \text{Atom}_\rho[\forall X. T] \mid \forall T_1, T_2, \forall R \in \text{Rel}[T_1, T_2]. (v_1 \ [T_1], v_2 \ [T_2]) \in \mathcal{J}_{\rho, X \mapsto (T_1, T_2, R)}[[T]]\}$$

where R relates values of types T_1 and T_2 , formally $\text{Rel}[T_1, T_2] = \{R \subseteq \text{Atom}[T_1, T_2]\}$. This relation allows us to relate values at abstract types: two values are related at an abstract type X , if they are in the relation for X :

$$\mathcal{V}_\rho[[X]] = \rho_R(X)$$

Finally, two computations are related if they reduce to two related values ($t \mapsto^* v$ specifies that term t reduces in zero or more steps to the value v).

$$\mathcal{J}_\rho[[T]] = \{(t_1, t_2) \in \text{Atom}_\rho[T] \mid t_1 \mapsto^* v_1 \Rightarrow (t_2 \mapsto^* v_2 \wedge (v_1, v_2) \in \mathcal{V}_\rho[[T]])\}$$

With the above definitions, we can establish the definition of the logical relation between two open terms. Two open terms are related if both are well-typed with the same type, and if we close them with any ρ and γ in the interpretation of Δ and Γ , respectively, we obtain related computations.

$$\Delta; \Gamma \vdash t_1 \leq t_2 : T \triangleq \Delta; \Gamma \vdash t_1 : T \wedge \Delta; \Gamma \vdash t_2 : T \wedge \forall \rho, \gamma. \rho \in \mathcal{D}[\Delta] \wedge \gamma \in \mathcal{G}_\rho[\Gamma] \Rightarrow (\rho(\gamma_1(t_1)), \rho(\gamma_2(t_2))) \in \mathcal{J}_\rho[[G]]$$

The fundamental property of this relation establishes that a well-typed program is related with itself: if $\Delta; \Gamma \vdash t : T$ then $\Delta; \Gamma \vdash t \preceq t : T$. The proof of this property uses compatibility lemmas for each term constructor. For instance, the compatibility lemma for type instantiation states: if $\Delta; \Gamma \vdash t_1 \preceq t_2 : \forall X.T$ and $\Delta \vdash T'$, then $\Delta; \Gamma \vdash t_1 [T'] \preceq t_2 [T'] : T[T'/X]$. An important property that is used to demonstrate parametricity (specifically the above compatibility lemma) (Ahmed, 2006; Ahmed et al., 2017) is the following (hereafter called *compositionality*):

$$\text{If } \Delta \vdash T', \rho \in \mathcal{D}[\Delta] \text{ and } R = \mathcal{V}_\rho \llbracket T' \rrbracket, \text{ then } \mathcal{V}_{\rho, X \mapsto (\rho_1(T'), \rho_2(T'), R)} \llbracket T \rrbracket = \mathcal{V}_\rho \llbracket T[T'/X] \rrbracket$$

Observe that for compositionality to be satisfied, the relation R can not be any relation; it must be $\mathcal{V}_\rho \llbracket T' \rrbracket$. For example, if $T = X$ and $T' = \text{Int}$, then:

$$\mathcal{V}_{\rho, X \mapsto (\rho_1(T'), \rho_2(T'), R)} \llbracket T \rrbracket = \mathcal{V}_{\rho, X \mapsto (\rho_1(\text{Int}), \rho_2(\text{Int}), R)} \llbracket X \rrbracket = \rho_R(X) = R$$

and

$$\mathcal{V}_\rho \llbracket T[T'/X] \rrbracket = \mathcal{V}_\rho \llbracket X[\text{Int}/X] \rrbracket = \mathcal{V}_\rho \llbracket \text{Int} \rrbracket$$

Therefore, $R = \mathcal{V}_\rho \llbracket T' \rrbracket = \mathcal{V}_\rho \llbracket \text{Int} \rrbracket$.

Parametricity for gradual languages—or *gradual parametricity*—is a novel concept around which different efforts have been developed, yielding different notions. The subtle differences in interpretation come from the specificities of gradual typing, namely the potential for runtime errors due to type imprecision. Much of it is linked to the mechanism used to enforce type abstraction. Apart from GSF, gradual parametricity has only been proven for λB (Ahmed et al., 2017) and PolyG $^\nu$ (New et al., 2020), under two fairly different interpretations. Technically, both are defined using logical relations that are fairly standard, except for three important cases: polymorphic types, type variables, and of course, the unknown type. We now briefly review and compare both approaches.

Gradual parametricity in λB . We now present the notion of gradual parametricity for λB , illustrating the main differences with the original notion of parametricity of Reynolds (1983). As we will see later, GSF follows similar ideas and techniques. Building on prior work by Matthews and Ahmed (2008), λB uses runtime type generation to reduce type applications, and a form of automatic (un)sealing is introduced via *conversions* and type names during reduction. A conversion $T_1 \xrightarrow{\phi} T_2$ is used to make explicit the conversion between a type name and the type it is bound to in the store. The label ϕ stands for a type name α accompanied by a sign ($-$ or $+$), where $-\alpha$ represents a sealing and $+\alpha$ an unsealing. For instance, $1 : \text{Int} \xrightarrow{-\alpha} \alpha$ is a conversion, representing an integer sealed value with type α . The term $(1 : \text{Int} \xrightarrow{-\alpha} \alpha) : \alpha \xrightarrow{+\alpha} \text{Int}$ is composed of two conversions, reducing to the plain value 1 after unsealing. Conversions are introduced upon type applications, similar to the ε_{out} evidence in GSF. They are also closely related to sealing and unsealing terms in PolyG $^\nu$.

Since gradual types introduce divergence, λB uses a *step-indexed* logical relation to ensure well-foundedness. Technically, this means that atoms in λB are of the form (W, t_1, t_2) , where world W describes the set of assumptions under which the pair of expressions t_1 and t_2 are related. Because reduction occurs relative to a type name store, and type names have indefinite dynamic extent, worlds are of the form $(j, \Sigma_1, \Sigma_2, \kappa)$: j corresponds to the step index, Σ_1 , and Σ_2 correspond to the type name stores under which the terms are being typechecked and evaluated, and κ is a map from type names to relations R . As worlds carry type instantiation information and relations, environment ρ now maps type variables to type names. For instance, $\rho(X) = (T_1, T_2, R)$ may correspond to $\rho(X) = \alpha$, in a world W such that $W = (j, \{\alpha := T_1\}, \{\alpha := T_2\}, \{\alpha \mapsto R\})$. For simplicity, we use a dot notation to access different components of a world: $W.j, W.\Sigma_1, W.\Sigma_2$, and

$W.\kappa$ are used to access the step-index, both type name stores, and the relation store, respectively. Note that the index $W.j$ specifies the number of available *future* reduction steps, *i.e.* a single reduction step reduces the index by one.

Two values are related at a type name α if both values are conversions to α and belong to the relation associated with α :

$$\mathcal{V}_\rho[\alpha] = \{(W, v_1 : T_1 \xrightarrow{\alpha} \alpha, v_2 : T_2 \xrightarrow{\alpha} \alpha) \in \text{Atom}_\emptyset[\alpha] \mid (\downarrow W, v_1, v_2) \in W.\kappa(\alpha)\}$$

Above, $\downarrow W$ lowers the step-index of the world and the interpretation κ in the world by one; formally $\downarrow W = (j, W.\Sigma_1, W.\Sigma_2, [W.\kappa]_j)$ and $j = W.j - 1$, where $[\kappa]_j = \{\alpha \mapsto [R]_j \mid \kappa(\alpha) = R\}$, and $[R]_j = \{(W, t_1, t_2) \in R \mid W.j < j\}$. For instance, $(W, 1 : \text{Int} \xrightarrow{\alpha} \alpha, 2 : \text{Int} \xrightarrow{\alpha} \alpha) \in \mathcal{V}_\rho[\alpha] = \mathcal{V}_\rho[X]$, when $\rho(X) = \alpha$, $W.\Sigma_1(\alpha) = \text{Int}$, $W.\Sigma_2(\alpha) = \text{Int}$, and $(\downarrow W, 1, 2) \in W.\kappa(\alpha)$.

As sealing and unsealing are introduced automatically at runtime, to reason parametrically about type abstractions, λB does not directly relate type applications as computations. For instance, consider term $\Lambda X.\lambda x : X.x$, which is related with itself $(W, \Lambda X.\lambda x : X.x, \Lambda X.\lambda x : X.x) \in \mathcal{V}_\emptyset[\forall X.X \rightarrow X]$. If we instantiate these values with Int , choosing relation $\{(1, 2)\}$, as $W.\Sigma_i \triangleright (\Lambda X.\lambda x : X.x) [\text{Int}] \mapsto W.\Sigma_i, \alpha := \text{Int} \triangleright (\lambda x : \alpha.x) : \alpha \rightarrow \alpha \xrightarrow{\alpha} \text{Int} \rightarrow \text{Int}$, then according to the standard definition of parametricity, the two instantiations have to be related at type $X \rightarrow X$. Following the dynamic semantics of λB , the following must hold:

$$(W', (\lambda x : \alpha.x) : \alpha \rightarrow \alpha \xrightarrow{\alpha} \text{Int} \rightarrow \text{Int}, (\lambda x : \alpha.x) : \alpha \rightarrow \alpha \xrightarrow{\alpha} \text{Int} \rightarrow \text{Int}) \in \mathcal{V}_\rho[X \rightarrow X]$$

for a future world W' such that $(W', 1, 2) \in W'.\kappa(\alpha)$, for some W'' . A future world intuitively captures how the world changes upon reduction: while the step-index decreases by one at each step of reduction, the store is extended after each type instantiation. Formally, we say that W' is a future world of W , notation $W' \succeq W$, if the step index is lower ($W'.j < W.j$), the type name stores are super sets of the originals ($W'.\Sigma_1 \supseteq W.\Sigma_1$ and $W'.\Sigma_2 \supseteq W.\Sigma_2$), and the $W'.\kappa$ is a future relation store ($W'.\kappa \succeq [W.\kappa]_{W'.j}$). We say that κ' is a future relation store of κ , notation $\kappa' \succeq \kappa$, if $\forall \alpha \in \text{dom}(\kappa)$ then $\kappa'(\alpha) = \kappa(\alpha)$. According to the definition of related functions at type $X \rightarrow X$, the application of these functions to related values at type X should yield related computations at type X . In particular using $(W', 1 : \text{Int} \xrightarrow{\alpha} \alpha, 2 : \text{Int} \xrightarrow{\alpha} \alpha) \in \mathcal{V}_\rho[X]$, then

$$\begin{aligned} & (W', ((\lambda x : \alpha.x) : \alpha \rightarrow \alpha \xrightarrow{\alpha} \text{Int} \rightarrow \text{Int}) (1 : \text{Int} \xrightarrow{\alpha} \alpha), \\ & ((\lambda x : \alpha.x) : \alpha \rightarrow \alpha \xrightarrow{\alpha} \text{Int} \rightarrow \text{Int}) (2 : \text{Int} \xrightarrow{\alpha} \alpha)) \in \mathcal{J}_\rho[X] \end{aligned}$$

But these application expressions do not type check! Therefore, instead of relating the two type application expressions as computations, λB relates only the bodies of the type abstractions *after* the type applications have been performed (highlighted in gray):

$$\begin{aligned} \mathcal{V}_\rho[\forall X.T] &= \{(W, v_1, v_2) \in \text{Atom}_\rho[\forall X.T] \mid \forall T_1, T_2, \forall R \in \text{Rel}[T_1, T_2]. \forall W' \succeq W. \forall \alpha. \forall t_1, t_2. \\ & W'.\Sigma_1 \triangleright v_1 [T_1] \mapsto W'.\Sigma_1, \alpha := T_1 \triangleright (t_1 : \rho(T)[\alpha/X] \xrightarrow{\alpha} \rho(T)[T_1/X]) \wedge \\ & W'.\Sigma_2 \triangleright v_2 [T_2] \mapsto W'.\Sigma_2, \alpha := T_2 \triangleright (t_2 : \rho(T)[\alpha/X] \xrightarrow{\alpha} \rho(T)[T_2/X]) \wedge \\ & (W' \boxtimes (\alpha, T_1, T_2, R), \mathbf{t_1, t_2}) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[T]\} \end{aligned}$$

After both type applications take a step, only the inner terms t_1 and t_2 are related in a world extended with α , the two instantiated types T_1 and T_2 , and the chosen relation R . World extension \boxtimes is formally defined as $W \boxtimes (\alpha, T_1, T_2, R) = (W.j, (W.\Sigma_1, \alpha := T_1), (W.\Sigma_2, \alpha := T_2), W.\kappa[\alpha \mapsto R])$.

Observe how this definition strips out the outermost conversions in charge of sealing and unsealing (this conversion is similar to evidence ε_{out} in GSF). This technique makes it possible to reason about a pair of related functions applied to a pair of already-sealed related values. In the previous example, we know that if $W.\Sigma_i \triangleright (\Lambda X.\lambda x : X.x) [\text{Int}] \mapsto W.\Sigma_i, \alpha := \text{Int} \triangleright (\lambda x : \alpha.x) : \alpha \rightarrow \alpha \stackrel{+\alpha}{\Rightarrow} \text{Int} \rightarrow \text{Int}$ then $(W', \lambda x : \alpha.x, \lambda x : \alpha.x) \in \mathcal{V}_\rho \llbracket X \rightarrow X \rrbracket$, where W' is the extended future world. Therefore we can deduce that $(W', (\lambda x : \alpha.x) (1 : \text{Int} \stackrel{-\alpha}{\Rightarrow} \alpha), (\lambda x : \alpha.x) (2 : \text{Int} \stackrel{-\alpha}{\Rightarrow} \alpha)) \in \mathcal{J}_{\rho[X \mapsto \alpha]} \llbracket X \rrbracket$.

The problem with this definition is that it does not support directly reasoning about type applications. For instance, in the previous example, from the logical relation we cannot directly deduce that:

$$(W', (\lambda x : \alpha.x) : \alpha \rightarrow \alpha \stackrel{+\alpha}{\Rightarrow} \text{Int} \rightarrow \text{Int}, (\lambda x : \alpha.x) : \alpha \rightarrow \alpha \stackrel{+\alpha}{\Rightarrow} \text{Int} \rightarrow \text{Int}) \in \mathcal{V}_\rho \llbracket \text{Int} \rightarrow \text{Int} \rrbracket$$

To reason about related type applications as computations (and not by considering the inner terms only), one needs to use a *conversion* lemma. This lemma relates two values after the unsealing of some type name α . Essentially, such a lemma says that if $(W, v_1, v_2) \in \mathcal{V}_\rho \llbracket T \rrbracket$, T and T' are convertible under $+\alpha$, $W.\Sigma_1(\alpha) = W.\Sigma_2(\alpha) = T''$ and $W.\kappa(\alpha) = \mathcal{V}_\rho \llbracket T'' \rrbracket$, then

$$(W, v_1 : \rho(T) \stackrel{+\alpha}{\Rightarrow} \rho(T'), v_2 : \rho(T) \stackrel{+\alpha}{\Rightarrow} \rho(T)) \in \mathcal{J}_\rho \llbracket T' \rrbracket$$

Observe that to apply this lemma, α must be bound to the same type in both stores ($W.\Sigma_1(\alpha) = W.\Sigma_2(\alpha) = T''$) and to the value relation of that type ($W.\kappa(\alpha) = \mathcal{V}_\rho \llbracket T'' \rrbracket$). When this situation happens, we say that α is *synchronized* in W . A similar requirement is established for the proof of parametricity for System F, specifically the compositionality lemma described before. The synchronization requirement is needed in gradual parametricity, among other reasons, to prevent the unsealing of unrelated values such as $(W', 1 : \text{Int} \stackrel{-\alpha}{\Rightarrow} \alpha, 2 : \text{Int} \stackrel{-\alpha}{\Rightarrow} \alpha) \in \mathcal{V}_\rho \llbracket \alpha \rrbracket$. Otherwise, after unsealing, we would have $(W'', 1, 2) \in \mathcal{V}_\rho \llbracket \text{Int} \rrbracket$, which is false.

Gradual parametricity in PolyG $^\nu$. New et al. (2020) recently developed another approach to gradual parametricity, which has the benefit of avoiding the convoluted treatment of type applications described above. In doing so, the notion of gradual parametricity they present is more similar to Reynolds's original presentation. Note however that this comes at a cost: the *syntax* of PolyG $^\nu$ departs importantly from System F, by requiring all sealing and unsealing to happen explicitly in the term syntax, with outward scoping of type variables:

$$\boxed{\text{System F}} \ ((\Lambda X.\lambda x : X.x) [\text{Int}] 1) + 1 \quad \boxed{\text{PolyG}^\nu} \ \text{unseal}_X((\Lambda X.\lambda x : X.x) [X = \text{Int}] (\text{seal}_X 1)) + 1$$

Technically, gradual parametricity for PolyG $^\nu$ is established by first translating PolyG $^\nu$ to an intermediate language PolyC $^\nu$ and finally to CBPV $_{\text{OSum}}$, a variant of Levy's Call-by-Push-Value (Levy, 1999) with open sums to encode the unknown type. The logical relation of parametricity is defined for CBPV $_{\text{OSum}}$, and differs importantly from that of λB . In particular, even though it still uses type names to relate type abstractions, the definition requires type applications (and not some inner terms) to be related as computations, as expected in the standard treatment of parametricity. Crucially, this is possible only because type applications never incur automatic insertion of conversions to seal/unseal values, as would happen in λB , because in this approach, sealing and unsealing are explicit in the syntax of terms.

Comparing parametricities. The notion of gradual parametricity of PolyG $^\nu$ is stronger than that of λB , as it directly embodies the kind of parametric reasoning that one is used to in static languages. While λB ensures a form of gradual parametricity, this notion is weaker, because given

two related type abstractions and two arbitrary (possibly different) types, we cannot directly reason about both corresponding type applications directly: we can only directly reason about the body of the type abstractions after the type applications have reduced.

While the notion of gradual parametricity of PolyG^ν is superior, as already mentioned, it is enabled by sacrificing the syntax of System F. In this work, we are interested in gradualizing System F, and studying the properties we can get, rather than in designing a different static source language in order to accommodate the desired reasoning principles. This led us to embrace runtime sealing through type names, as in λB , and consequently, to aspire to a weaker notion of gradual parametricity than that of PolyG^ν . We do believe that both approaches are fully valuable and necessary to understand the many ways in which gradual typing can embrace such an advanced typing discipline.

In particular, as illustrated by New et al. (2020), the weaker notion of parametricity adopted in GSF can lead to behavior that breaks the (strong notion of) parametricity enjoyed by PolyG^ν . Note that this can however only occur when manipulating values of *imprecise* polymorphic types; for values of static types, the reasoning principles of standard parametricity do apply. They argue that GSF exhibits non-parametric behavior by considering the following example. Consider the below value:

$$v \triangleq (\lambda X. \lambda x : X. \text{true}) :: \forall X. ? \rightarrow \text{Bool}$$

Although v is related to itself at type $\forall X. ? \rightarrow \text{Bool}$, two *different* instantiations (to Int and Bool , respectively) are not related computations, *i.e.* $(W, v [\text{Int}], v [\text{Bool}]) \notin \mathcal{T}_\rho[? \rightarrow \text{Bool}]$. Given two related arguments at type $?$ such as $\varepsilon_{\text{Int}}3 :: ?$ twice, $v [\text{Int}] (\varepsilon_{\text{Int}}3 :: ?)$ reduces to true , whereas $v [\text{Bool}] (\varepsilon_{\text{Int}}3 :: ?)$ reduces to an error. As we saw earlier, this happens because GSF does not directly relate type applications as computations. The logical relation only tells us that after instantiation, the internal terms (without the outermost evidences) $\varepsilon_{\alpha^{\text{Int}} \rightarrow \text{Bool}}(\lambda x : \alpha. \text{true}) :: ? \rightarrow \text{Bool}$ and $\varepsilon_{\alpha^{\text{Bool}} \rightarrow \text{Bool}}(\lambda x : \alpha. \text{true}) :: ? \rightarrow \text{Bool}$ are indeed related at type $? \rightarrow \text{Bool}$. In this case, if we try to apply both functions to $\varepsilon_{\text{Int}}3 :: ?$, both programs fail. The only arguments that can be passed such that both applications succeed are related sealed values at type $?$, such as $(W, \langle \text{Int}, \alpha^{\text{Int}} \rangle 3 :: ?, \langle \text{Int}, \alpha^{\text{Bool}} \rangle \text{true} :: ?) \in \mathcal{V}_\rho[?]$ (assuming an appropriate relation for α).

Finally, note that the fact that $v [\text{Bool}] (1 :: ?)$ reduces to an error in GSF points to a wider point in the design space of gradually-typed languages: how *eagerly* should type constraints be checked? Indeed, $v [\text{Bool}]$ is $\lambda x : \text{Bool}. \text{true}$, whose application to an underlying Int value is ill-typed and can legitimately be expected to fail. In that respect, GSF follows GTLC (Siek and Taha, 2006; Siek et al., 2015a), in which $(\lambda x : \text{Bool}. \text{true}) (1 :: ?)$ also fails with a runtime cast error. This eager form of runtime type checking likewise follows from the Abstracting Gradual Typing methodology as formulated by Garcia et al. (2016). An interesting perspective would be to study a lazy variant of AGT (where casts or evidence are accumulated in a value and are only reduced when the value is used), and whether it recovers properties of alternative approaches (New and Ahmed, 2018).

It is interesting to observe that no runtime error is raised in λB for this example, despite the fact that the parametricity logical relation is essentially the same as that of GSF. The difference comes from the runtime semantics of λB : as we have illustrated in §2.4, λB does not track the type instantiations that occur on imprecise types. This means that the underlying typing violation observed by GSF, which manifests as a runtime error, is not noticed in λB . Therefore, this example highlights yet another point of tension in the design space of System F-based gradual languages.

3.7.2 Gradual Parametricity in GSF

We now turn to the technical details of gradual parametricity in GSF. As explained above, we follow λB (Ahmed et al., 2017) for the formal development of gradual parametricity, due to the use of runtime type name generation for sealing, and the System F syntax that requires automatic insertion of (un)sealing evidences at runtime. We highlight the main differences in the logical relations of GSF with respect to λB , mainly in the value logical relations for types $?$ and α .

We establish parametricity for GSF by proving parametricity for GSF_ε . Specifically, we define a *step-indexed* logical relation for GSF_ε terms, closely following the relation for λB . The relation is defined on atoms (W, t_1, t_2) that denote two related terms t_1, t_2 in a world W . A world is composed of a step index j , two stores Ξ_1 and Ξ_2 used to typecheck and evaluate the related terms, and a mapping κ , which maps type names to relations R , used to relate sealed values. The components of a world are accessed through a dot notation, *e.g.* $W.j$ for the step index. The interpretations of values, terms, stores, name environments, and type environments are mutually defined, using the auxiliary definitions of Figure 3.14. As usual, the value and term interpretations are indexed by a type and a type substitution ρ .

Auxiliary definitions. We write $\text{Atom}_\rho[G]$ (Figure 3.14) to denote a set of terms of the same type after substitution. The $\text{Atom}_\rho^\equiv[G]$ is similar to $\text{Atom}_\rho[G]$ but restricts the set to values that have, after substitution, equally precise evidences (the equality is after unlifting because two sealed values may be related under different instantiations). Remember that the unlifting operator, given an evidence type E , returns a gradual type G , forgetting to which types the type names were instantiated. For instance,

$$(W, \langle \text{Int}, \alpha^{\text{Int}} \rangle 1 :: \alpha, \langle \text{Bool}, \alpha^{\text{Bool}} \rangle \text{true} :: \alpha) \in \text{Atom}_\rho^\equiv[X]$$

if $\rho(X) = \alpha$, as $(W, \langle \text{Int}, \alpha^{\text{Int}} \rangle 1 :: \alpha, \langle \text{Bool}, \alpha^{\text{Bool}} \rangle \text{true} :: \alpha) \in \text{Atom}_\rho[X]$ (assuming an adequate world) and $\text{unlift}(\pi_2(\langle \text{Int}, \alpha^{\text{Int}} \rangle)) = \text{unlift}(\pi_2(\langle \text{Bool}, \alpha^{\text{Bool}} \rangle)) = \alpha$. However,

$$(W, \langle \text{Int}, \text{Int} \rangle 1 :: ?, \langle \text{Bool}, \text{Bool} \rangle \text{true} :: ?) \notin \text{Atom}_\rho^\equiv[?]$$

since $\text{unlift}(\pi_2(\langle \text{Int}, \text{Int} \rangle)) = \text{Int} \neq \text{Bool} = \text{unlift}(\pi_2(\langle \text{Bool}, \text{Bool} \rangle))$. We explain this in detail below, when presenting the logical relations for values.

$\text{REL}_n[G_1, G_2]$ defines the set of relations of values of type G_1 and G_2 . Note that if $R \in \text{REL}_n[G_1, G_2]$, we also require that for all atoms in R , all future versions of that atom should also be present in R . Intuitively, this is because values in a relation R should still be related after lowering the number of steps (reduction). We use $\lfloor R \rfloor_n$ and $\lfloor \kappa \rfloor_n$ to restrict the step index of the worlds to less than n . Finally, $\kappa' \succeq \kappa$ specifies that κ' is a future relation mapping of κ (an extension⁸), and similarly $W' \succeq W$ expresses that W' is a future world of W . Intuitively, a future relation mapping represents the same relations as the original plus some extra ones that may have been added during reduction. Similarly, a future world represents a world after some steps of reduction, *i.e.* a world with a smaller (or equal) step index and a future mapping relation.

Logical relation for terms. Following λB , the logical interpretation of terms (Figure 3.15) of a given type enforces an “error-sensitive” view of parametricity: if the first term yields a value, the second must produce a related value at that type; if the first term fails, so must the second.

⁸Note the relation is antisymmetric as we quantify over all $\alpha \in \text{dom}(\kappa)$, and this could be false for some $\alpha \in \text{dom}(\kappa')$.

$$\begin{aligned}
\text{Atom}_n[G_1, G_2] &= \{(W, t_1, t_2) \mid W.j < n \wedge W \in \text{WORLD}_n \wedge W.\Xi_1, \cdot, \cdot \vdash t_1 : G_1 \wedge W.\Xi_2, \cdot, \cdot \vdash t_2 : G_2\} \\
\text{Atom}_n^{\text{val}}[G_1, G_2] &= \{(W, v_1, v_2) \in \text{Atom}_n[G_1, G_2]\} \quad \text{Atom}_\rho[G] = \cup_{n \geq 0} \{(W, t_1, t_2) \in \text{Atom}_n[\rho(G), \rho(G)]\} \\
\text{Atom}_\rho^{\bar{}}[G] &= \{(W, v_1, v_2) \in \text{Atom}_\rho[G] \mid \text{unlift}(\pi_2(\text{ev}(v_1))) = \text{unlift}(\pi_2(\text{ev}(v_2)))\} \\
\text{WORLD} &= \cup_{n \geq 0} \text{WORLD}_n \\
\text{WORLD}_n &= \{(j, \Xi_1, \Xi_2, \kappa) \in \text{NAT} \times \text{STORE} \times \text{STORE} \times (\text{TYPERNAME} \rightarrow \text{REL}_j) \mid \\
&\quad j < n \wedge \vdash \Xi_1 \wedge \vdash \Xi_2 \wedge \forall \alpha \in \text{dom}(\kappa). \kappa(\alpha) \in \text{REL}_j[\Xi_1(\alpha), \Xi_2(\alpha)]\} \\
\text{REL}_n[G_1, G_2] &= \{R \in \text{Atom}_n^{\text{val}}[G_1, G_2] \mid \forall (W, v_1, v_2) \in R. \forall W' \succeq W. (W', v_1, v_2) \in R\} \\
[R]_n &= \{(W, t_1, t_2) \in R \mid W.j < n\} \quad [\kappa]_n = \{\alpha \mapsto [R]_n \mid \kappa(\alpha) = R\} \\
\kappa' \succeq \kappa &\triangleq \forall \alpha \in \text{dom}(\kappa). \kappa'(\alpha) = \kappa(\alpha) \\
W' \succeq W &\triangleq W'.j \leq W.j \wedge W'.\Xi_1 \supseteq W.\Xi_1 \wedge W'.\Xi_2 \supseteq W.\Xi_2 \wedge W'.\kappa \succeq [W.\kappa]_{W'.j} \wedge W', W \in \text{WORLD} \\
\downarrow W &= (j, W.\Xi_1, W.\Xi_2, [W.\kappa]_j) \quad \text{where } j = W.j - 1 \\
W \boxtimes (\alpha, G_1, G_2, R) &= (W.j, (W.\Xi_1, \alpha := G_1), (W.\Xi_2, \alpha := G_2), W.\kappa[\alpha \mapsto R])
\end{aligned}$$

Figure 3.14: Logical Relation: Auxiliary Definitions

The reason behind this is to ensure parametric behavior in the presence of runtime errors. Given a parametric term, if after two different instantiations one of the resulting terms fails and the other terminates to a value, then both instantiations did not behave similarly.⁹ Observe that one reduction takes i steps in the definition of the interpretation of terms while the other one takes any arbitrary number of steps. This is because only one index is needed for this definition to be well-founded, and it would be challenging to establish the number of steps for the second reduction.

Logical relations for values. The logical interpretation of values (Figure 3.15) uses $\text{Atom}_\rho^{\bar{}}[G]$, which requires the second component of the evidence of each value to have the same precision to enforce such sensitivity. Indeed, if one is allowed to be more precise than the other, then when later combined in the same context, the more precise value may induce failure while the other does not. For instance, if we have evidences $\langle \text{Int} \rightarrow \text{Int}, \alpha^{\text{Int} \rightarrow \text{Int}} \rangle$ and $\langle \text{Int} \rightarrow \text{Int}, \text{Int} \rightarrow \text{Int} \rangle$ and combine them through consistent transitivity with the evidence $\langle \text{Int} \rightarrow \text{Int}, \text{Int} \rightarrow \text{Int} \rangle$, the first combination fails while the second one succeeds, resulting in $\langle \text{Int} \rightarrow \text{Int}, \text{Int} \rightarrow \text{Int} \rangle$. For this reason, related values are required to have evidence such that their second components are equal using the unlifting operator ($\text{unlift}(\pi_2(\langle \text{Int} \rightarrow \text{Int}, \alpha^{\text{Int} \rightarrow \text{Int}} \rangle)) = \alpha \neq \text{Int} \rightarrow \text{Int} = \text{unlift}(\pi_2(\langle \text{Int} \rightarrow \text{Int}, \text{Int} \rightarrow \text{Int} \rangle))$).

Two base values are related if they are equal. Two functions are related if their application to related values yields related results. Note that, unlike λB , the arguments are related at one step down ($\downarrow W$).¹⁰ Otherwise (in combination with the definition of related values at type $?$, which also presents some differences), the logical relation would not be well-founded, as we explain below. Two pairs are related if their components are pointwise related. Two type abstractions are related if given any two types and any relation between them, the instantiated terms (without their unsealing evidence) are also related in a world extended (\boxtimes) with α , the two instantiation types G_1 and G_2 and the chosen relation R between sealed values (Figure 3.14). Note that the step index of this extended world is decreased by one, because we take a reduction step.

Two sealed values are related at a type name α if first, after unsealing, the resulting values are

⁹ Considering a logical relation where the first term may fail and the other terminates leads to a weaker version of parametricity (in terms of different notions of gradual parametricity).

¹⁰ The operator $\downarrow W$ has the same definition as $\blacktriangleright W$ in λB .

$\mathcal{V}_\rho[B]$	$= \{(W, v, v) \in \text{Atom}_\rho^-[B]\}$		
$\mathcal{V}_\rho[G_1 \rightarrow G_2]$	$= \{(W, v_1, v_2) \in \text{Atom}_\rho^-[G_1 \rightarrow G_2] \mid \forall W' \succeq W. \forall v'_1, v'_2. (\Downarrow W', v'_1, v'_2) \in \mathcal{V}_\rho[G_1] \Rightarrow (W', v_1, v_2) \in \mathcal{V}_\rho[G_2]\}$		
$\mathcal{V}_\rho[G_1 \times G_2]$	$= \{(W, v_1, v_2) \in \text{Atom}_\rho^-[G_1 \times G_2] \mid (W, \pi_1(v_1), \pi_1(v_2)) \in \mathcal{V}_\rho[G_1] \wedge (W, \pi_2(v_1), \pi_2(v_2)) \in \mathcal{V}_\rho[G_2]\}$		
$\mathcal{V}_\rho[\forall X.G]$	$= \{(W, v_1, v_2) \in \text{Atom}_\rho^-[\forall X.G] \mid \forall W' \succeq W. \forall t_1, t_2, G_1, G_2, \alpha, \varepsilon_1, \varepsilon_2. \forall R \in \text{REL}_{W',j}[G_1, G_2].$ $(W'.\Xi_1 \vdash G_1 \wedge W'.\Xi_2 \vdash G_2 \wedge$ $W'.\Xi_1 \triangleright v_1[G_1] \mapsto W'.\Xi_1, \alpha := G_1 \triangleright \varepsilon_1 t_1 :: \rho(G)[G_1/X] \wedge$ $W'.\Xi_2 \triangleright v_2[G_2] \mapsto W'.\Xi_2, \alpha := G_2 \triangleright \varepsilon_2 t_2 :: \rho(G)[G_2/X]) \Rightarrow$ $(\Downarrow W' \boxtimes (\alpha, G_1, G_2, R), t_1, t_2) \in \mathcal{V}_\rho[X \mapsto \alpha][G]\}$		
$\mathcal{V}_\rho[X]$	$= \mathcal{V}_\rho[\rho(X)]$		
$\mathcal{V}_\rho[\alpha]$	$= \{(W, \langle E_{11}, \alpha^{E_{12}} \rangle_{u_1} :: \alpha, \langle E_{21}, \alpha^{E_{22}} \rangle_{u_2} :: \alpha) \in \text{Atom}_\emptyset^-[\alpha] \mid$ $(\Downarrow W, \langle E_{11}, E_{12} \rangle_{u_1} :: W.\Xi_1(\alpha), \langle E_{21}, E_{22} \rangle_{u_2} :: W.\Xi_2(\alpha)) \in W.\kappa(\alpha) \wedge$ $(\forall \Xi, \varepsilon, G. (W \in \mathcal{S}[\Xi] \wedge \varepsilon \Vdash \Xi \vdash \alpha \sim G) \Rightarrow$ $(W, \varepsilon(\langle E_{11}, \alpha^{E_{12}} \rangle_{u_1} :: \alpha) :: G, \varepsilon(\langle E_{21}, \alpha^{E_{22}} \rangle_{u_2} :: \alpha) :: G) \in \mathcal{V}_\rho[G])\}$		
$\mathcal{V}_\rho[?]$	$= \{(W, \varepsilon_1 u_1 :: ?, \varepsilon_2 u_2 :: ?) \in \text{Atom}_\emptyset^-[?] \mid \text{const}(\pi_2(\varepsilon_i)) = G \wedge$ $(W, \varepsilon_1 u_1 :: G, \varepsilon_2 u_2 :: G) \in \mathcal{V}_\rho[G]\}$		
$\mathcal{J}_\rho[G]$	$= \{(W, t_1, t_2) \in \text{Atom}_\rho[G] \mid \forall i < W.j. (\forall \Xi_1, v_1. W.\Xi_1 \triangleright t_1 \mapsto^i \Xi_1 \triangleright v_1 \Rightarrow$ $\exists W' \succeq W, v_2. W.\Xi_2 \triangleright t_2 \mapsto^* W'.\Xi_2 \triangleright v_2 \wedge W'.j + i = W.j \wedge$ $W'.\Xi_1 = \Xi_1 \wedge (W', v_1, v_2) \in \mathcal{V}_\rho[G]) \wedge$ $(\forall \Xi_1. W.\Xi_1 \triangleright t_1 \mapsto^i \mathbf{error} \Rightarrow \exists \Xi_2. W.\Xi_2 \triangleright t_2 \mapsto^* \mathbf{error})\}$		
$\mathcal{S}[\cdot]$	$= \text{WORLD}$		
$\mathcal{S}[\Xi, \alpha := G]$	$= \mathcal{S}[\Xi] \cap \{W \in \text{WORLD} \mid W.\Xi_1(\alpha) = G \wedge W.\Xi_2(\alpha) = G \wedge$ $\vdash W.\Xi_1 \wedge \vdash W.\Xi_2 \wedge W.\kappa(\alpha) = \lfloor \mathcal{V}_\emptyset[G] \rfloor_{W,j}\}$		
$\mathcal{D}[\cdot]$	$= \{(W, \emptyset) \mid W \in \text{WORLD}\}$		
$\mathcal{D}[\Delta, X]$	$= \{(W, \rho[X \mapsto \alpha]) \mid (W, \rho) \in \mathcal{D}[\Delta] \wedge \alpha \in \text{dom}(W.\kappa)\}$		
$\mathcal{G}_\rho[\cdot]$	$= \{(W, \emptyset) \mid W \in \text{WORLD}\}$		
$\mathcal{G}_\rho[\Gamma, x : G]$	$= \{(W, \gamma[x \mapsto (v_1, v_2)]) \mid (W, \gamma) \in \mathcal{G}_\rho[\Gamma] \wedge (W, v_1, v_2) \in \mathcal{V}_\rho[G]\}$		
$\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : G \triangleq$	$\Xi; \Delta; \Gamma \vdash t_1 : G \wedge \Xi; \Delta; \Gamma \vdash t_2 : G \wedge \forall W \in \mathcal{S}[\Xi], \rho, \gamma.$ $((W, \rho) \in \mathcal{D}[\Delta] \wedge (W, \gamma) \in \mathcal{G}_\rho[\Gamma]) \Rightarrow (W, \rho(\gamma_1(t_1)), \rho(\gamma_2(t_2))) \in \mathcal{V}_\rho[G]$		
$\Xi; \Delta; \Gamma \vdash t_1 \approx t_2 : G \triangleq$	$\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : G \wedge \Xi; \Delta; \Gamma \vdash t_2 \preceq t_1 : G$		
$\text{const}(B) = B$	$\text{const}(\alpha^G) = \alpha$	$\text{const}(E_1 \rightarrow E_2) = ? \rightarrow ?$	$\text{const}(\forall X.E) = \forall X.?$
$\text{const}(E_1 \times E_2) = ? \times ?$			

Figure 3.15: Gradual Logical Relation

in the relation corresponding to α ($W.\kappa(\alpha)$) in a one step lower current world. The first part of the definition is faithful to λB , while the second part is new. We additionally require that for any evidence ε that justifies the judgment between α and any type G , in any store such that W belongs to its interpretation, the values ascribed to the type G and evidence ε remain related. This technical extension is sufficient to prove Lemma 3.32 (formalized at the end of this section), which states that the ascription of two related values yields related terms; this lemma is essential for the proof of parametricity. The necessity of this extension comes from differences between the dynamic semantics

of GSF and λB . The dynamic semantics of GSF combine evidence (eager), whereas λB accumulates cast (lazy). For instance, the conversion $(1 : \text{Int} \xrightarrow{-\alpha} \alpha) : \alpha \xrightarrow{-\beta} \beta$ from λB needs two reduction steps to obtain 1, *i.e.* the step-index is reduced by two. In GSF, this information is compressed in a single evidence $\langle \text{Int}, \beta^{\alpha^{\text{Int}}} \rangle$, and needs only one reduction step to obtain the similar value, *i.e.* the step-index is reduced by one. Observe that if ε exists such that $\varepsilon \Vdash \Xi \vdash \alpha \sim G$, since $W \in \mathcal{S}[\Xi]$, α must be synchronized (*i.e.* $W.\Sigma_1(\alpha) = W.\Sigma_2(\alpha) = G'$ and $W.\kappa(\alpha) = \mathcal{V}_\rho[\![G']\!]$). Intuitively, if α is synchronized, then after multiple possible unsealings the resulting values are kept related. For instance, if $(W, \langle \text{Int}, \alpha^{\beta^{\text{Int}}} \rangle 1 :: \alpha, \langle \text{Int}, \alpha^{\beta^{\text{Int}}} \rangle 1 :: \alpha) \in \mathcal{V}_\rho[\![\alpha]\!]$ where α is synchronized, then it must be the case that $(W', \langle \text{Int}, \beta^{\text{Int}}} \rangle 1 :: \beta, \langle \text{Int}, \beta^{\text{Int}}} \rangle 1 :: \beta) \in W.\kappa(\beta)$, and also that $(W'', \langle \text{Int}, \text{Int} \rangle 1 :: \text{Int}, \langle \text{Int}, \text{Int} \rangle 1 :: \text{Int}) \in \mathcal{V}_\rho[\![\text{Int}]\!]$; but if $(W, \langle \text{Int}, \alpha^{\beta^{\text{Int}}} \rangle 1 :: \alpha, \langle \text{Bool}, \alpha^{\beta^{\text{Bool}}} \rangle \text{true} :: \alpha) \in \mathcal{V}_\rho[\![\alpha]\!]$, then it must be the case that $(W', \langle \text{Int}, \beta^{\text{Int}}} \rangle 1 :: \text{Int}, \langle \text{Bool}, \beta^{\text{Bool}}} \rangle \text{true} :: \text{Bool}) \in W.\kappa(\beta)$.

Finally, two values are related at type $?$ if they are related at the least-precise type with the same top-level constructor as the second component of the evidence, $\text{const}(\pi_2(\varepsilon_i))$. The function const extracts the top-level constructor of an evidence type (Figure 3.15). The intuition is that to be able to relate these unknown values we must take a step towards relating their actual content; evidence necessarily captures at least the top-level constructor (*e.g.* if a value is a function, the second evidence type is no less precise than $? \rightarrow ?$, *i.e.* $\text{const}(E_1 \rightarrow E_2)$). Also, we consider the second component as in $\langle E_1, E_2 \rangle \Vdash \Xi; \Delta \vdash G \sim G'$, E_1 and E_2 correspond to the most precise information about G and G' respectively. Here E_2 corresponds to the most precise information about $?$, and E_1 could be a totally unrelated type as in $(W, \langle \text{Int}, \alpha^{\text{Int}}} \rangle 1 :: ?, \langle \text{Bool}, \alpha^{\text{Bool}}} \rangle \text{true} :: ?) \in \mathcal{V}_\rho[\![?]\!]$. Observe that, unlike λB , the definition does not decrease the index of the world W by 1. However, in λB this is done on a case-by-case basis where needed (*i.e.*, function, type name, etc.). We made this technical change to facilitate the parametricity proof.

Well-foundedness. The logical relation is well-founded for three reasons: (i) in the $?$ case, $\text{const}(\pi_2(\varepsilon_i))$ cannot itself be $?$, as just explained; (ii) in each other recursive cases, the step index is lowered: for functions and pairs, the relation is between reducible expressions (applications, projections) that either take a step or fail; for type abstractions, the relation is with respect to a world whose index is lowered; (iii) we require in the definition of related functions that arguments must be related at (at least) one step down. For instance, if $(W, \langle ? \rightarrow ?, ? \rightarrow ? \rangle u_1 :: ?, \langle ? \rightarrow ?, ? \rightarrow ? \rangle u_2 :: ?) \in \mathcal{V}_\rho[\![?]\!]$, then $(W, \langle ? \rightarrow ?, ? \rightarrow ? \rangle u_1 :: ? \rightarrow ?, \langle ? \rightarrow ?, ? \rightarrow ? \rangle u_2 :: ? \rightarrow ?) \in \mathcal{V}_\rho[\![? \rightarrow ?]\!]$, but function arguments related at $\mathcal{V}_\rho[\![?]\!]$ would contain the original $(W, \langle ? \rightarrow ?, ? \rightarrow ? \rangle u_1 :: ?, \langle ? \rightarrow ?, ? \rightarrow ? \rangle u_2 :: ?)$ atom.

Logical relations for stores and environments. The unary relation $\mathcal{S}[\Xi]$ specifies all the worlds that satisfy Ξ : every α in $\text{dom}(\Xi)$ must be synchronized. The interpretation of $\mathcal{D}[\Delta]$ specifies all pairs of worlds W and type substitutions ρ , such that all type variables in Δ are mapped to some α in ρ , and α is associated to some relation in W . The relation $\mathcal{G}_\rho[\![\Gamma]\!]$ specifies that the value environment γ satisfies the type environment Γ under world W if, for every variable $x \in \text{dom}(\Gamma)$, the mapped values are related in $\mathcal{V}_\rho[\![\Gamma(x)]\!]$ in world W .

For convenience, we introduce the following notation for projections in γ : if $\gamma = \{x \mapsto (v_{11}, v_{12}), y \mapsto (v_{21}, v_{22}), \dots\}$, then $\gamma_1 = \{x \mapsto v_{11}, y \mapsto v_{21}, \dots\}$ and $\gamma_2 = \{x \mapsto v_{12}, y \mapsto v_{22}, \dots\}$. Type variable substitution $\rho_i(s)$ is defined as syntactic sugar for $\rho(W.\Xi_i, s)$, in a context where W is defined, lifting each substituted type name in the process and defined as

$$\begin{aligned} (\rho, X \mapsto \alpha)(\Xi, s) &= \rho(\Xi, s[\text{lift}_\Xi(\alpha)/X]) \\ \cdot(\Xi, s) &= s \end{aligned}$$

Parametricity. The logical relation approximation $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : G$ says that given a world W that satisfies Ξ , a type substitution ρ and value environment γ that satisfies Δ and Γ respectively under world W , then the pair of substituted terms $\rho_1(\gamma_1(t_1)), \rho_2(\gamma_2(t_2))$ are related computations in $\mathcal{J}_\rho[[G]]$. Logical equivalence $\Xi; \Delta; \Gamma \vdash t_1 \approx t_2 : G$ is defined as the symmetric extension of logical approximation. Finally, the fundamental property says that any well-typed GSF ε term is related to itself at its type:

Theorem 3.28 (Fundamental Property) *If $\Xi; \Delta; \Gamma \vdash t : G$ then $\Xi; \Delta; \Gamma \vdash t \preceq t : G$.*

As standard, the proof of the fundamental property uses compatibility lemmas for each term constructor and the compositionality lemma. The compatibility lemmas related to type abstractions are the following:

Lemma 3.29 (Compatibility-E Λ) *If $\Xi; \Delta, X \vdash t_1 \preceq t_2 : G$, $\varepsilon \vdash \Xi; \Delta \vdash \forall X. G \sim G'$ and $\Xi; \Delta \vdash \Gamma$ then $\Xi; \Delta; \Gamma \vdash \varepsilon(\Lambda X. t_1) :: G' \preceq \varepsilon(\Lambda X. t_2) :: G' : G'$.*

Lemma 3.30 (Compatibility-EappG) *If $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : \forall X. G$ and $\Xi; \Delta \vdash G'$, then $\Xi; \Delta; \Gamma \vdash t_1 [G'] \preceq t_2 [G'] : G[G'/X]$.*

The compatibility lemma for type abstractions (Lemma 3.29) says that if two terms are related, then the type abstractions (whose bodies are those terms) ascribed to any type G' are also related at G' . The compatibility lemma for type instantiations (Lemma 3.30) says that if two terms are related to some polymorphic type $\forall X. G$, then the instantiations to some type G' are related at $G[G'/X]$. The remaining compatibility lemmas are defined analogously and can be found in the Appendix.

In order to prove Lemma 3.30 (Compatibility-EappG), we establish another important lemma to relate terms after a type substitution.

Lemma 3.31 (Compositionality) *If*

- $W.\Xi_i(\alpha) = \rho(G')$ and $W.\kappa(\alpha) = \nu_\rho[[G']]$,
- $E'_i = \text{lift}_{W.\Xi_i}(\rho(G'))$,
- $E_i = \text{lift}_{W.\Xi_i}(G_p)$ for some $G_p \sqsubseteq \rho(G)$,
- $\rho' = \rho[X \mapsto \alpha]$,
- $\varepsilon_i = \langle E_i[\alpha^{E'_i}/X], E_i[E'_i/X] \rangle$, such that $\varepsilon_i \vdash W.\Xi_i \vdash \rho(G[\alpha/X]) \sim \rho(G[G'/X])$, and
- $\varepsilon_i^{-1} = \langle E_i[E'_i/X], E_i[\alpha^{E'_i}/X] \rangle$, such that $\varepsilon_i^{-1} \vdash W.\Xi_i \vdash \rho(G[G'/X]) \sim \rho(G[\alpha/X])$, then

1. $(W, v_1, v_2) \in \nu_\rho[[G]] \Rightarrow (W, \varepsilon_1 v_1 :: \rho(G[G'/X]), \varepsilon_2 v_2 :: \rho(G[G'/X])) \in \mathcal{J}_\rho[[G[G'/X]]]$
2. $(W, v_1, v_2) \in \nu_\rho[[G[G'/X]]] \Rightarrow (W, \varepsilon_1^{-1} v_1 :: \rho'(G), \varepsilon_2^{-1} v_2 :: \rho'(G)) \in \mathcal{J}_{\rho'}[[G]]$

Observe that Lemma 3.31 is informally the combination of the compositionality and conversion lemma from Ahmed et al. (2017). This lemma says that if α is synchronized in W , and X is bound to α , then (1) given two related values at type G , removing variable X by substitution (unsealing)

yields related computations at type $G[G'/X]$; and (2) given two related values at type $G[G'/X]$ then substituting G' for X (sealing) yields related computations at type G . Note that the un(sealing) of type names is done via ascriptions, where evidences are constructed as in the reduction rule for type instantiations.

Almost all compatibility lemmas and the compositionality lemma rely on the fact that the ascription of two related values yields related terms.

Lemma 3.32 (Ascriptions Preserve Relations) *If $(W, v_1, v_2) \in \mathcal{V}_\rho[[G]]$, $\varepsilon \Vdash \Xi$; $\Delta \vdash G \sim G'$, $W \in \mathcal{S}[[\Xi]]$, and $(W, \rho) \in \mathcal{D}[[\Delta]]$, then $(W, \rho_1(\varepsilon)v_1 :: \rho(G'), \rho_2(\varepsilon)v_2 :: \rho(G')) \in \mathcal{T}_\rho[[G']]$.*

3.7.3 Parametricity vs. the DGG in GSF

We now give a different perspective from that presented in §3.6.1, regarding the violation of the general dynamic gradual guarantee (DGG, stated with respect to \sqsubseteq). More precisely, we show that the definition of parametricity for GSF (§3.7) is incompatible with the DGG. To do so, we again prove that there exists two terms in GSF, related by precision, whose behavior violates the DGG, but this time we do so with a proof of the intermediate results that is fully-driven by the definition of parametricity, and not by monotonicity of consistent transitivity. We present the proof sketch of the intermediate results in order to highlight the key properties that imply this incompatibility. This is particularly relevant because these properties also manifest in λB , for which the DGG has not been formally explored yet.

Recall from §3.6.1 that the term that helps us establish the violation of the DGG is $id_?$, a variant of the polymorphic identity function id_X whose term variable x is given the unknown type. This term always fails when fully applied.

Lemma 3.33 *For any $\vdash v : ?$ and $\vdash G$, we have $(\Lambda X. \lambda x : ?. x :: X) [G] v \Downarrow \mathbf{error}$.*

Let us consider $id_? \triangleq \Lambda X. \lambda x : ?. x :: X$, whose elaboration is $v_a = \langle \forall X. ? \rightarrow X, \forall X. ? \rightarrow X \rangle (\Lambda X. \lambda x : ?. \langle \alpha^{Int}, \alpha^{Int} \rangle x :: \alpha) :: ? \rightarrow Int$ and two different types Int and $Bool$, such that:

$$\begin{aligned} \cdot \triangleright v_a [Int] \mapsto \\ \alpha := Int \triangleright \langle ? \rightarrow \alpha^{Int}, ? \rightarrow Int \rangle (\langle ? \rightarrow \alpha^{Int}, ? \rightarrow \alpha^{Int} \rangle (\lambda x : ?. \langle \alpha^{Int}, \alpha^{Int} \rangle x :: \alpha) :: ? \rightarrow \alpha) :: ? \rightarrow Int \end{aligned}$$

and

$$\begin{aligned} \cdot \triangleright v_a [Bool] \mapsto \\ \alpha := Bool \triangleright \langle ? \rightarrow \alpha^{Bool}, ? \rightarrow Bool \rangle (\langle ? \rightarrow \alpha^{Bool}, ? \rightarrow \alpha^{Bool} \rangle (\lambda x : ?. \langle \alpha^{Bool}, \alpha^{Bool} \rangle x :: \alpha) :: ? \rightarrow \alpha) :: ? \rightarrow Bool \end{aligned}$$

If we consider the domain of the external evidences:

$$dom(\langle ? \rightarrow \alpha^{Int}, ? \rightarrow Int \rangle) = dom(\langle ? \rightarrow \alpha^{Bool}, ? \rightarrow Bool \rangle) = \langle ?, ? \rangle$$

and any pair of related values at type $?$, then their ascription to $?$ using evidence $\langle ?, ? \rangle$ yields related values at type $?$. In particular for $v_b = \langle Int, Int \rangle 1 :: ?$, $(W, v_b, v_b) \in \mathcal{V}[[?]]$, as $\langle Int, Int \rangle \S \langle ?, ? \rangle = \langle Int, Int \rangle (\langle ?, ? \rangle v_b :: ? \mapsto v_b)$, we obtain that $(\downarrow W \boxtimes (\alpha, Int, Bool, R), v_b, v_b) \in \mathcal{V}_{X \mapsto \alpha}[[?]]$ for any $R \in \mathbf{REL}_{W,j}[Int, Bool]$. This fact is central to the proof of Lemma 3.33, and to prove it, it is used

a (rather technical) intermediate lemma, which crisply captures this idea in a more general setting using the term $(\Lambda X.\lambda x : ?.t)$, where t can be any term. Intuitively, consider one step of execution of the application of $(\Lambda X.\lambda x : ?.t)$ to two different arbitrary types, and the resulting outermost evidences ε_1 and ε_2 . The ascription of any value v to $?$ using the domain of ε_1 and ε_2 yields two related computations. The intermediate lemma is formalized as follows:

Lemma 3.34 *Let $\vdash (\Lambda X.\lambda x : ?.t) \rightsquigarrow v_a : \forall X. ? \rightarrow X$ and $\vdash v \rightsquigarrow v_b : ?$. Let G_1 and G_2 , such that $\text{const}(G_1) \neq \text{const}(G_2)$. If $\cdot \triangleright v_a [G_i] \mapsto \alpha := G_i \triangleright \varepsilon_i v_i :: ? \rightarrow G_i$ and $\varepsilon_i \Vdash ? \rightarrow \alpha \sim ? \rightarrow G_i$, then $\forall W \in \mathcal{S}[\cdot], \forall R \in \text{REL}_{W,j}[G_1, G_2], (W \boxtimes (\alpha, G_1, G_2, R), \text{dom}(\varepsilon_1)v_b :: ?, \text{dom}(\varepsilon_2)v_b :: ?) \in \mathcal{F}_{X \mapsto \alpha}[\cdot]$.*

We now show that instantiating $id_?$ to any arbitrary type such as Int , and applying it to any value of type $?$ such as $v_b = \langle \text{Int}, \text{Int} \rangle 1 :: ?$ necessarily leads to a runtime error (Lemma 3.33). For simplicity, we omit worlds in the development below. Consider the configuration $\cdot \triangleright v_a [\text{Int}] v_b$, which steps to:

$$\alpha := \text{Int} \triangleright (\langle ? \rightarrow \alpha^{\text{Int}}, ? \rightarrow \text{Int} \rangle (\langle ? \rightarrow \alpha^{\text{Int}}, ? \rightarrow \alpha^{\text{Int}} \rangle (\lambda x : ?. \langle \alpha^{\text{Int}}, \alpha^{\text{Int}} \rangle x :: \alpha) :: ? \rightarrow \alpha) :: ? \rightarrow \text{Int}) v_b$$

By the fundamental property (Th. 3.28) on $id_?$ and 1, we know that:

1. v_a is related to itself at type $\forall X. ? \rightarrow X$, and then choosing $G_1 = \text{Int}, G_2 = \text{Bool}, R = \{(\langle \text{Int}, \text{Int} \rangle 1 :: \text{Int}, \langle \text{Bool}, \text{Bool} \rangle \text{true} :: \text{Bool})\}$ we know that

$$(v_1, v_2) \in \mathcal{V}_{X \mapsto \alpha}[\cdot \rightarrow \alpha]$$

where $v_1 = \langle ? \rightarrow \alpha^{\text{Int}}, ? \rightarrow \alpha^{\text{Int}} \rangle (\lambda x : ?. \langle \alpha^{\text{Int}}, \alpha^{\text{Int}} \rangle x :: \alpha) :: ? \rightarrow \alpha$, and $v_2 = \langle ? \rightarrow \alpha^{\text{Bool}}, ? \rightarrow \alpha^{\text{Bool}} \rangle (\lambda x : ?. \langle \alpha^{\text{Bool}}, \alpha^{\text{Bool}} \rangle x :: \alpha) :: ? \rightarrow \alpha$.

2. $\langle \text{Int}, \text{Int} \rangle 1 :: ?$ is related to itself at type $?$.

Then we notice that, by associativity of consistent transitivity, the pending redex is equivalent to:

$$\alpha := \text{Int} \triangleright \langle \alpha^{\text{Int}}, \text{Int} \rangle (v_1 (\langle ?, ? \rangle v_b :: ?)) :: \text{Int}$$

By (2) and Lemma 3.34, $\langle ?, ? \rangle v_b :: ? \mapsto v_b$ and $(v_b, v_b) \in \mathcal{V}_{X \mapsto \alpha}[\cdot]$. We instantiate the result in (1) $(v_1, v_2) \in \mathcal{V}_{X \mapsto \alpha}[\cdot \rightarrow \alpha]$ with arguments $(v_b, v_b) \in \mathcal{V}_{X \mapsto \alpha}[\cdot]$. But notice that $v_2 v_b$ always fails (as α is instantiated to Bool not Int), therefore $v_1 v_b$ must also fail and the result holds, otherwise v_1 and v_2 would not be related. Furthermore, let us assume (falsely) that $v_2 v_b$ reduces to some value. Then $\alpha := \text{Int} \triangleright v_1 v_b \mapsto^* \Xi \triangleright \langle \text{Int}, \alpha^{\text{Int}} \rangle 1 :: \alpha$, and $\alpha := \text{Int} \triangleright v_2 v_b \mapsto^* \Xi \triangleright \langle \text{Int}, \alpha^{\text{Int}} \rangle 1 :: \alpha$, then we would have to prove that $(\langle \text{Int}, \alpha^{\text{Int}} \rangle 1 :: \alpha, \langle \text{Int}, \alpha^{\text{Int}} \rangle 1 :: \alpha) \in \mathcal{V}_{X \mapsto \alpha}[\alpha]$, i.e. $(\langle \text{Int}, \text{Int} \rangle 1 :: \text{Int}, \langle \text{Int}, \text{Int} \rangle 1 :: \text{Int}) \in R$ which is false as $R = \{(\langle \text{Int}, \text{Int} \rangle 1 :: \text{Int}, \langle \text{Bool}, \text{Bool} \rangle \text{true} :: \text{Bool})\}$. Therefore $v_1 v_b$ ought to fail.

As a consequence of Lemma 3.33, the dynamic gradual guarantee is violated in GSF.

Corollary 3.35 *There exist $\vdash t_1 : G$ and t_2 , such that $t_1 \sqsubseteq t_2$, $t_1 \Downarrow v$ and $t_2 \Downarrow \text{error}$.*

PROOF. Let $id_X \triangleq \Lambda X.\lambda x : X.x :: X$, and $id_? \triangleq \Lambda X.\lambda x : ?.x :: X$. By definition of precision, we have $id_X \sqsubseteq id_?$. Let $\vdash v : G$ and $\vdash v' : ?$, such that $v \sqsubseteq v'$. Pose $t_1 \triangleq id_X [G] v$ and $t_2 \triangleq id_? [G] v'$. By definition of precision, we have $t_1 \sqsubseteq t_2$. By evaluation, $t_1 \Downarrow v$. But by Lemma 3.33, $t_2 \Downarrow \text{error}$. \square

Interestingly, Lemma 3.33 holds irrespective of the actual choices for representing evidence in $\text{GSF}\varepsilon$. The key reason is the logical interpretation of $\forall X.G$. Therefore, we conjecture that the incompatibility described here does not only apply to GSF but to other gradual languages that use similar logical relations, such as λB .

3.7.4 Gradual Free Theorems in GSF

The parametricity logical relation (§3.7) allows us to define notions of logical approximation (\preceq) and equivalence (\approx) that are sound with respect to contextual approximation (\preceq^{ctx}) and equivalence (\approx^{ctx}), and hence can be used to derive free theorems about well-typed GSF terms (Wadler, 1989; Ahmed et al., 2017). The definitions of contextual approximation and equivalence, and the soundness of the logical relation, are fairly standard. As shown by Ahmed et al. (2017), in a gradual setting, the free theorems that hold for System F are weaker, as they have to be understood “modulo errors and divergence”. Ahmed et al. (2017) prove two such free theorems in λB . However, these free theorems only concern *fully static* type signatures. This leaves unanswered the question of what *imprecise* free theorems are enabled by gradual parametricity. To the best of our knowledge, this topic has not been formally developed in the literature so far, despite several claims about expected theorems, exposed hereafter.

Igarashi et al. (2017a) report that the System F polymorphic identity function, if allowed to be cast to $\forall X.? \rightarrow X$, would always trigger a runtime error when applied, suggesting that functions of type $\forall X.? \rightarrow X$ are always failing. Consequently, System F_G rejects such a cast by adjusting the precision relation (§2.4). But the corresponding free theorem—*i.e.* that applying any function of type $\forall X.? \rightarrow X$ either diverges or fails—is not proven. Also, Ahmed et al. (2011) declare that parametricity dictates that any value of type $\forall X.X \rightarrow ?$ is either constant or always failing or diverging (p.7). This gradual free theorem is not proven either. In fact, in both an older system (Ahmed et al., 2009b) and its newest version (Ahmed et al., 2017), as well as in System F_G , casting the identity function to $\forall X.X \rightarrow ?$ yields a function that returns *without errors*, though the returned value is still sealed, and as such is unusable (§2.4). The parametricity relation in GSF does not impose such behavior: it only imposes uniformity of behavior, including failure, of polymorphic terms, which leaves some freedom regarding when to fail. As we saw earlier, the unknown type can stand for any type, including any type variable. Consequently, in GSF, a function of type $\forall X.? \rightarrow X$ could behave like the identity function with type $\forall X.X \rightarrow X$, or as a function of type $\forall X.\text{Int} \rightarrow X$ that always fails when applied, or a function that given a pair returns its first or second component with type $\forall X.(X \times X) \rightarrow X$, etc. In particular, we show next that ascribing the System F identity function to $\forall X.? \rightarrow X$ yields a function that behaves exactly as the identity function (and hence never fails).

The DGG^{\leq} -related Lemmas 3.21 and 3.24 help us prove that in GSF types $\forall X.? \rightarrow X$ and $\forall X.X \rightarrow ?$ are inhabited by non-constant, non-failing, parametricity-preserving terms. Observe that this result is a consequence of the fact that imprecise ascriptions are harmless in GSF.

In particular, both types admit the ascribed System F identity function, among many others (for instance, the polymorphic term $\Lambda X.\lambda x : X.\lambda f : X \rightarrow X.f x$ of type $\forall X.X \rightarrow (X \rightarrow X) \rightarrow X$ can also be ascribed to $\forall X.X \rightarrow ?$).

We formalize this using the following corollary:

Corollary 3.36 *Let t and v be static terms such that $\vdash t : \forall X.T$, $\vdash v : T'$, and $t [T'] v \Downarrow v'$.*

1. If $\forall X.T \sqsubseteq \forall X.X \rightarrow ?$ then $(t :: \forall X.X \rightarrow ?) [T'] v \Downarrow v''$, and $v' \leq v''$.
2. If $\forall X.T \sqsubseteq \forall X.? \rightarrow X$ then $(t :: \forall X.? \rightarrow X) [T'] v \Downarrow v''$, and $v' \leq v''$.

Cheap Theorems. The intuition of $\forall X.? \rightarrow X$ denoting always-failing functions is not entirely misguided: in GSF, this result does hold *for a subset* of the terms of that type. This leads us to observe that we can derive “cheap theorems” with gradual parametricity: obtained not by looking only at the type, but by also considering the head constructors of a term. For instance:

Theorem 3.37 *Let $v \triangleq \Lambda X.\lambda x : ?.t$ for some t , such that $\vdash v : \forall X.? \rightarrow X$. Then for any $\vdash v' : G$, we either have $v [G] v' \Downarrow \mathbf{error}$ or $v [G] v' \Uparrow$.*

This result is proven by exploiting the gradual parametricity result (Theorem 3.28). Note that what makes it a “free” theorem is that it holds independently of the body t , therefore *without having to analyze the whole term*. Not as good as a free theorem, but cheap. It is worth noting that although the external loss of precision is harmless, the internal loss of precision may change the expected behavior of a term. For example, the function $\Lambda X.\lambda x : ?.t$ from Theorem 3.37 might be the imprecise identity function $\Lambda X.\lambda x : ?.x :: X$. Therefore, we could expect that applied to a type and a value of the same type, it returns the same value; however, by Theorem 3.37, it always fails or diverges.

3.8 Related Work

We have already discussed at length related work on gradual parametricity, especially the most recent developments (Ahmed et al., 2017; Igarashi et al., 2017a; Xie et al., 2018; New et al., 2020), highlighting the different design choices, properties and limitations of each. Hopefully our discussions adequately reflect the many subtleties involved in designing a gradual parametric language.

The relation between parametric polymorphism in general and dynamic typing much predates the work on gradual typing. Abadi et al. (1991) first note that without further precaution, type abstraction might be violated. Subsequent work explored different approaches to protect parametricity, especially runtime-type generation (RTG) (Leroy and Mauny, 1991; Abadi et al., 1995; Rossberg, 2003). Sumii and Pierce (2004) and Guha et al. (2007) use dynamic sealing, originally proposed by Morris (1973), in order to dynamically enforce type abstraction. Matthews and Ahmed (2008) also use RTG in order to protect polymorphic functions in an integration of Scheme and ML. This line of work eventually led to the polymorphic blame calculus (Ahmed et al., 2011) and its most recent version with the proof of parametricity by Ahmed et al. (2017). We adapt their logical relation to the evidence-based semantics of GSF.

Hou et al. (2016) prove the correctness of compiling polymorphism to dynamic typing with embeddings and partial projections; the compilation setting however differs significantly from gradual typing. New and Ahmed (2018) use embedding-projection pairs to formulate a semantic account of the dynamic gradual guarantee, coined *graduality*, in a language with explicit casts. Inspired by the work of Neis et al. (2009) on parametricity in a non-parametric language, they extended their approach to gradual parametricity, yielding the PolyG ^{ν} language design with explicit sealing (New et al., 2020), discussed at length in this chapter.

This work is generally related to gradualization of advanced typing disciplines, in particular

to gradual information-flow security typing (Disney and Flanagan, 2011; Fennell and Thiemann, 2013, 2016; Garcia and Tanter, 2015; Toro et al., 2018). In these systems, one aims at preserving *noninterference*, *i.e.* that private values do not affect public outputs. Both parametricity and noninterference are 2-safety properties, expressed as a relation of two program executions. While Garcia and Tanter (2015) show that one can derive a pure security language with AGT that satisfies both noninterference and the dynamic gradual guarantee, Toro et al. (2018) find that in the presence of mutable references, one can have either the dynamic gradual guarantee, or noninterference, but not both. Also similarly to this work, AGT for security typing Toro et al. (2018) needs a more precise abstraction for evidence types (based on security *label intervals*) in order to enforce noninterference. Together, these results suggest that type-based approaches to gradual typing are in tension with semantically-rich typing disciplines. Solutions might come from restricting the considered *syntax*, as in PolyG^ν in the context of parametricity, or the *range* of graduality, as recently established by Azevedo de Amorim et al. (2020) in the context of noninterference, where the dynamic end of the spectrum is not fully untyped security-wise.

3.9 Conclusion

GSF is a type-driven gradual parametric language, occurring sealing generation and resolution in its runtime semantics. This appears necessary in order to respect the syntax of System F, but implies that GSF violates the dynamic gradual guarantee in certain cases. We precisely characterize the weaker continuity that GSF supports, along with all its other properties. The design of GSF is largely driven by the Abstracting Gradual Typing (AGT) methodology. We find that AGT greatly streamlines the static semantics of GSF, but does not yield a language that respects parametricity by default; non-trivial exploration was necessary to uncover how to strengthen the structure and treatment of runtime evidence in order to recover a notion of gradual parametricity. In turn, this strengthening breaks the dynamic gradual guarantee when loss of precision interferes with type-driven sealing.

There are two main trends in the design of gradual parametric languages: those based on type-driven sealing, such as System F, like λB , CSA, System F_G, and GSF, and those that depart from that syntax, being based on term-driven sealing, like PolyG^ν. We believe that GSF goes beyond prior work based on the type-driven sealing approach, despite its lack of full support for the dynamic gradual guarantee. While PolyG^ν enjoys a stronger metatheory than languages from the other trend, several limitations regarding modularity and abstraction caused by explicit sealing/unsealing are not benign. Therefore, the question remains whether there is a third way to embrace both System F and a fully satisfying metatheory.

The following two chapters explore embedding an untyped lambda calculus with sealing/unsealing primitives into GSF (Chapter 4) and the extension of GSF with existential types (Chapter 5), which are the core of data abstraction mechanisms.

Chapter 4

Embedding Dynamic Sealing in GSF

A gradual language is expected to cover a spectrum between two typing disciplines, such as simple static typing and dynamic typing. The static end of the spectrum is characterized by the conservative extension results (Siek et al., 2015a), which we have established for GSF with respect to System F (Proposition 3.10 and Proposition 3.17). The dynamic end of the spectrum is typically characterized by an *embedding* from the considered dynamic language to the gradual language (Siek et al., 2015a). For instance, in the case of GTLC (Siek and Taha, 2006), the dynamic language is an untyped lambda calculus with primitives.

In this chapter, we study the “dynamic end” of GSF. Unsurprisingly, GSF can embed an untyped lambda calculus with primitives, called λ_{dyn} (§4.2). More interestingly, we highlight the expressive power of the underlying type name generation mechanism of GSF by proving that it can faithfully embed an untyped lambda calculus with *dynamic sealing*, λ_{seal} . This language, also known as the cryptographic lambda calculus, was first studied in a typed version by Pierce and Sumii (2000), and then untyped (Sumii and Pierce, 2004). One of their objectives was to study whether dynamic sealing could be used in order to dynamically impose parametricity via a compiler from System F to λ_{seal} . Recently, Devriese et al. (2018) prove that such a compiler is not fully abstract, *i.e.* compiled System F equivalent terms are not contextually equivalent in λ_{seal} . Nevertheless, the dynamic sealing mechanism of λ_{seal} to protect abstract data, and its relation to gradual parametricity, is an interesting question. We define an embedding of λ_{seal} terms into GSF (§4.4), and prove that this embedding is semantics preserving (§4.5).

4.1 Overview

In GSF, we can define a pair of functions of type $\forall X.(X \rightarrow ?) \times (? \rightarrow X)$ to emulate the runtime sealing primitives of the cryptographic lambda calculus. The first component of the pair is a function of type $X \rightarrow ?$, which intuitively justifies sealing the argument (of type X) at runtime, but not unsealing the returned value (of type $?$). Dually, the type of the second component is $? \rightarrow X$, which only justifies unsealing the returned value. The underlying mechanism ensures that the unsealing function only succeeds if its argument was sealed by the first function:

```
let p:  $\forall X.(X \rightarrow ?) \times (? \rightarrow X) = \Lambda X. \langle \lambda x:X. x::?, \lambda x:?. x::X \rangle$  in
let su = p [?] in
let seal = (fst su) in
```

```

let unseal = (snd su) in
(unseal (seal 1)) + 1    -----> 2

```

On the second line, `p [?]` creates a fresh pair of functions with an underlying type name that acts as the runtime sealing key: therefore, `seal` seals the value 1, and `unseal` unseals it; the whole program reduces to 2. Unsealing the sealed value with any other generated unsealing function, or attempting to add directly to the sealed value, yields a runtime error.

4.2 Embedding a Dynamically-Typed Language in GSF

The essence of embedding a dynamically-typed language in a gradual language is to ascribe every introduction form with the unknown type (Siek and Taha, 2006; Siek et al., 2015a). For instance the expression (1 2) from a dynamically-typed language can be embedded as (1 :: ?) (2 :: ?). Observe that not adding the ascriptions would yield an ill-typed term, as per the conservative extension result with respect to the static typing discipline. Let us call λ_{dyn} the dynamically-typed lambda calculus with pairs and primitives. We aim at an embedding of λ_{dyn} that preserves termination, divergence, and failure. We will establish this result formally as a corollary of a stronger result for the extended language with dynamic sealing (see Corollary 4.8).

The embedding of λ_{dyn} terms into GSF is defined as:

$$\begin{array}{ll}
\llbracket b \rrbracket = b :: ? & \llbracket x \rrbracket = x \\
\llbracket \lambda x.t \rrbracket = (\lambda x : ?. \llbracket t \rrbracket) :: ? & \llbracket t_1 t_2 \rrbracket = \text{let } x : ? = \llbracket t_1 \rrbracket \text{ in let } y : ? = \llbracket t_2 \rrbracket \text{ in } x y \\
\llbracket \langle t_1, t_2 \rangle \rrbracket = \langle \llbracket t_1 \rrbracket, \llbracket t_2 \rrbracket \rangle :: ? & \llbracket \pi_1(t) \rrbracket = \pi_1(\llbracket t \rrbracket) \\
\llbracket \text{op}(\bar{t}) \rrbracket = \text{let } \bar{x} : ? = \llbracket \bar{t} \rrbracket \text{ in } \text{op}(\bar{x}) :: ? & \llbracket \pi_2(t) \rrbracket = \pi_2(\llbracket t \rrbracket)
\end{array}$$

The only novelty here with respect to prior work is that the embedding produces application terms in A-normal form in order to ensure that embedded terms behave as expected. For example, the term (1 Ω), with $\Omega = (\lambda x.x x) (\lambda x.x x)$, diverges in the dynamically-typed language. But if we would embed an application $\llbracket t_1 t_2 \rrbracket$ simply as $\llbracket t_1 \rrbracket \llbracket t_2 \rrbracket$, evidence combination would detect the underlying type error before reducing the application. Note that this precaution is unnecessary for pairs, because there are no typing constraints between both components. To better understand the need to use the A-normal form, we present both translations (with and without A-normal form) of program (1 Ω) to GSF and then their translations to GSF_ε . For simplicity, let us suppose that $\llbracket \Omega \rrbracket = \Omega$ and, 1_ε and Ω_ε are GSF_ε terms, where $\vdash \Omega \rightsquigarrow \Omega_\varepsilon : ?$, $\vdash (1 :: ?) \rightsquigarrow 1_\varepsilon : ?$ and $1_\varepsilon = \varepsilon_{\text{Int}} 1 :: ?$.

$$\begin{array}{ll}
\llbracket t_1 t_2 \rrbracket = \text{let } x : ? = \llbracket t_1 \rrbracket \text{ in let } y : ? = \llbracket t_2 \rrbracket \text{ in } x y & \llbracket t_1 t_2 \rrbracket = \llbracket t_1 \rrbracket \llbracket t_2 \rrbracket \\
\text{GSF} \quad \text{let } x : ? = 1 :: ? \text{ in let } y : ? = \Omega \text{ in } x y & \text{GSF} \quad (1 :: ?) \Omega \\
\text{GSF}_\varepsilon \quad \text{let } x : ? = 1_\varepsilon \text{ in let } y : ? = \Omega_\varepsilon \text{ in } (\varepsilon_{? \rightarrow ?} x :: ? \rightarrow ?) y & \text{GSF}_\varepsilon \quad (\varepsilon_{? \rightarrow ?} 1_\varepsilon :: ? \rightarrow ?) \Omega_\varepsilon \\
\mapsto \cdot \triangleright \quad \text{let } y : ? = \Omega_\varepsilon \text{ in } (\varepsilon_{? \rightarrow ?} 1_\varepsilon :: ? \rightarrow ?) y & \mapsto \cdot \triangleright \quad \mathbf{error} \\
& \text{diverges reducing } \Omega_\varepsilon & (\varepsilon_{\text{Int}} \ ; \ \varepsilon_{? \rightarrow ?}) \text{ fails}
\end{array}$$

$$\begin{array}{lcl}
& x \in \text{VAR}, \sigma \in \text{SEAL}, \mu \subset \text{SEAL} & \\
t ::= & b \mid \lambda x.t \mid \langle t, t \rangle \mid x \mid t t \mid \pi_i(t) \mid \text{op}(\bar{t}) \mid \nu x.t \mid \{t\}_t \mid \text{let } \{x\}_t = t \text{ in } t \mid \sigma & \text{(terms)} \\
v ::= & b \mid \lambda x.t \mid \langle v, v \rangle \mid \{v\}_\sigma \mid \sigma & \text{(values)} \\
\text{error} ::= & \text{type_error} \mid \text{seal_type_error} \mid \text{unseal_error} & \text{(errors)}
\end{array}$$

$t \parallel \mu \longrightarrow t \parallel \mu$ or **error** **Notion of reduction**

$$\begin{array}{l}
(\lambda x.t) v \parallel \mu \longrightarrow t[v/x] \parallel \mu \quad \pi_i(\langle v_1, v_2 \rangle) \parallel \mu \longrightarrow v_i \parallel \mu \quad \text{op}(\bar{v}) \parallel \mu \longrightarrow \delta(\text{op}, \bar{v}) \parallel \mu \\
\nu x.t \parallel \mu \longrightarrow t[\sigma/x] \parallel \mu, \sigma \quad \text{where } \sigma \notin \mu \quad \{v\}_{v'} \parallel \mu \longrightarrow \text{seal_type_error} \quad \text{where } \not\exists \sigma. v' \equiv \sigma \\
\text{let } \{x\}_\sigma = \{v\}_{\sigma'} \text{ in } t \parallel \mu \longrightarrow \begin{cases} t[v/x] \parallel \mu & \sigma \equiv \sigma' \\ \text{unseal_error} & \sigma \not\equiv \sigma' \end{cases} \\
\text{let } \{x\}_\sigma = v \text{ in } t \parallel \mu \longrightarrow \text{seal_type_error} \quad \text{where } \not\exists \sigma'. v' . v \equiv \{v'\}_{\sigma'} \\
\text{let } \{x\}_v = v' \text{ in } t \parallel \mu \longrightarrow \text{seal_type_error} \quad \text{where } \not\exists \sigma. v \equiv \sigma
\end{array}$$

$t \parallel \mu \mapsto t \parallel \mu$ or **error** **Evaluation frames and reduction**

$$\begin{array}{l}
f ::= \square t \mid v \square \mid \langle \square, t \rangle \mid \langle v, \square \rangle \mid \pi_i(\square) \mid \{\square\}_t \mid \{v\}_\square \mid \text{op}(\bar{v}, \square, \bar{t}) \quad \text{(term frames)} \\
\mid \text{let } \{x\}_\square = t \text{ in } t \mid \text{let } \{x\}_v = \square \text{ in } t \\
\frac{t \parallel \mu \longrightarrow t' \parallel \mu'}{t \parallel \mu \mapsto t' \parallel \mu'} \qquad \frac{t \parallel \mu \mapsto t' \parallel \mu'}{f[t] \parallel \mu \mapsto f[t'] \parallel \mu'} \\
\frac{t \parallel \mu \longrightarrow \text{error}}{t \parallel \mu \mapsto \text{error}} \qquad \frac{t \parallel \mu \mapsto \text{error}}{f[t] \parallel \mu \mapsto \text{error}}
\end{array}$$

Figure 4.1: λ_{seal} : Untyped Lambda Calculus with Sealing

4.3 The Cryptographic Lambda Calculus λ_{seal}

The cryptographic lambda calculus λ_{seal} is an extension of λ_{dyn} with primitives for protecting abstract data by sealing (Sumii and Pierce, 2004). Figure 4.1 presents the syntax and dynamic semantics of the λ_{seal} language we consider here, which is a simplified variant of that of Sumii and Pierce (2004). In addition to standard terms, which correspond to λ_{dyn} , the λ_{seal} language introduces four new syntactic constructs dedicated to sealing. First, the term $\nu x.t$ generates a fresh key to seal and unseal values, bound to x in the body t . Seals, denoted by the metavariable σ , are values tracked in the set of allocated seals μ . The sealing construct $\{t_1\}_{t_2}$ evaluates t_1 to a value v and t_2 to a seal σ , and seals v with σ . Term $\text{let } \{x\}_{t_1} = t_2 \text{ in } t$ is for unsealing. At runtime, t_1 should evaluate to a seal σ and t_2 to a sealed value $\{v\}_{\sigma'}$. If $\sigma = \sigma'$, unsealing succeeds and t is evaluated with x bound to v . Otherwise, unsealing fails, producing a runtime sealing error **unseal_error**.¹

¹ The original term for unsealing in λ_{seal} has the syntax $\text{let } \{x\}_{t_1} = t_2 \text{ in } t \text{ else } t_3$; if the unsealing fails, reduction recovers from error evaluating t_3 . To be able to encode such a construct, we would need to extend GSF with error handling.

To illustrate, consider the following term:

$$\nu x.\nu y.\lambda b.\text{let } \{n\}_x = \{1\}_{(\text{if } b \text{ then } x \text{ else } y)} \text{ in } n + 1$$

This term first generates two fresh seals x and y , and then defines a function that receives a boolean b and attempts to unseal a sealed value. The value 1 is sealed using either x or y , depending on b , and unsealed with x . If the function is applied to `true`, unsealing succeeds because the seals coincide, and the function returns 2 . Otherwise, unsealing fails, and an `unseal_error` is raised.

Overall, we can distinguish three kinds of runtime errors, grouped in `error`, in λ_{seal} (Section 4.5 explain why we introduce this distinction). In addition to unsealing errors, `unseal_error`, there are two kinds of runtime *type* errors, hereafter called `type_error` and `seal_type_error`. The former, omitted in Figure 4.1 for simplicity, corresponds to standard runtime type errors such as applying a non-function, and can happen in λ_{dyn} . The latter is specific to λ_{seal} and corresponds to expressions that do not produce seals when expected or misuse seals, such as $\{1\}_2$. We omit for simplicity the errors related to the incorrect use of the seals, for example, the application of a seal in function position, such as $(\sigma \ 42)$. It is important to note that the correctness of the type system of the typed version of λ_{seal} with respect to the semantics guarantees that a well-typed program can not incur either a `type_error` or a `seal_type_error`, just an `unseal_error`.

4.4 Embedding λ_{seal} in GSF

We now present a semantic-preserving embedding of λ_{seal} terms into GSF. The embedding relies on a general seal/unseal generator, expressed as a GSF term. This term, called su hereafter, is a polymorphic pair of two functions, of type $\forall X.(X \rightarrow ?) \times (? \rightarrow X)$, instantiated at the unknown type, and ascribed to the unknown type:

$$su \equiv (\Lambda X. \langle (\lambda x : X.x :: ?), (\lambda x : ?.x :: X) \rangle) [?] :: ?$$

When evaluated, the type application generates a fresh type name, simulating the seal generation of λ_{seal} 's term $\nu x.t$. Then the first component of the pair represents a sealing function, while the second component represents an unsealing function, which can only successfully be applied to values sealed with the first component. We write su^σ to denote a particular value resulting from the evaluation of the term su , where the type name σ is generated and stored in Ξ . Crucially, a value that passed through $\pi_1(su^\sigma)$ is sealed, and can only be observed after passing through the unsealing function $\pi_2(su^\sigma)$. Trying to unseal it with a different function results in a runtime error.

Embedding Translation. Figure 4.2 defines the embedding from λ_{seal} to GSF. The cases unrelated to sealing are as presented in §4.2. The crux of the embedding is in the use of the term su . A seal generation term $\nu x.t$ is embedded into GSF by let-binding the variable x to the term su , whose value su^σ will be substituted in the translation of t . Recall that the first component of the pair su^σ is used for sealing, and the second one for unsealing. Therefore, the sealing operation $\{t_1\}_{t_2}$ is embedded by let-binding the translations of t_1 and t_2 to fresh variables x and y , and applying the first component of y (the sealing function) to x (the value to be sealed). Likewise, an unsealing `let $\{z\}_{t_1} = t_2$ in t_3` is embedded by binding the translation of t_1 and t_2 to fresh variables x and y , then unsealing y using the second component of x (the unsealing function), and binding the result to z , for use in the translation of the term t_3 . The use of A-normal forms in the embedding of

$$\begin{array}{ll}
\llbracket b \rrbracket = b :: ? & \llbracket x \rrbracket = x \\
\llbracket \lambda x.t \rrbracket = (\lambda x : ?. \llbracket t \rrbracket) :: ? & \llbracket t_1 t_2 \rrbracket = \text{let } x : ? = \llbracket t_1 \rrbracket \text{ in let } y : ? = \llbracket t_2 \rrbracket \text{ in } x y \\
\llbracket \langle t_1, t_2 \rangle \rrbracket = \langle \llbracket t_1 \rrbracket, \llbracket t_2 \rrbracket \rangle :: ? & \llbracket \pi_1(t) \rrbracket = \pi_1(\llbracket t \rrbracket) \\
\llbracket \text{op}(\bar{t}) \rrbracket = \text{let } \bar{x} : ? = \llbracket \bar{t} \rrbracket \text{ in } \text{op}(\bar{x}) :: ? & \llbracket \pi_2(t) \rrbracket = \pi_2(\llbracket t \rrbracket) \\
\\
\llbracket \nu x.t \rrbracket = \text{let } x : ? = su \text{ in } \llbracket t \rrbracket \\
\llbracket \{t_1\}_{t_2} \rrbracket = \text{let } x : ? = \llbracket t_1 \rrbracket \text{ in let } y : ? = \llbracket t_2 \rrbracket \text{ in } \pi_1(y) x \\
\llbracket \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 \rrbracket = \text{let } x : ? = \llbracket t_1 \rrbracket \text{ in let } y : ? = \llbracket t_2 \rrbracket \text{ in let } z : ? = \pi_2(x) y \text{ in } \llbracket t_3 \rrbracket \\
\text{where } su \equiv ((\Lambda X. ((\lambda x : X.x :: ?), (\lambda x : ?.x :: X))) [?]) :: ?
\end{array}$$

Figure 4.2: Embedding λ_{seal} in GSF

sealing and unsealing is required because both are eventually interpreted as function applications, so the precaution discussed in §4.2 applies. Finally, note that because seals σ cannot appear in the source language, the translation need not consider them.

Illustration. As example, the embedding of the λ_{seal} term $\nu x.\nu y.\text{let } \{n\}_x = \{1\}_x \text{ in } n + 1$ is the following GSF term:

```

let x : ? = su in
let y : ? = su in
let u : ? = x in
let z : ? = (let n1 : ? = 1 in let s : ? = x in π1(s) n1) in
let n : ? = π2(u) z in n + 1

```

The following reduction trace shows the most critical steps of the program above. We define su_ε as the translation of su to GSF ε , and su_ε^σ is the reduced value from su_ε , where a fresh seal σ is generated. Note that we omit some trivial evidences and type annotations for readability. This program generates two fresh type names (σ and σ'), reducing the first su to su_ε^σ and the second one to $su_\varepsilon^{\sigma'}$. Then, after a few substitution steps, the first component of su_ε^σ is applied to 1, sealing the value, and then applies the second component of su_ε^σ , unsealing the sealed value. The whole program reduces to 2.

	·	▷ let $x = su_\varepsilon$ in let $y = su_\varepsilon$ in ...	initial program
→*	$\sigma := ?, \sigma' := ?$	▷ let $u = su_\varepsilon^\sigma$ in ... let $s = su_\varepsilon^{\sigma'}$ in ...	σ and σ' are generated
→*	$\sigma := ?, \sigma' := ?$	▷ let $z = \pi_1(su_\varepsilon^\sigma)(\varepsilon_{\text{Int}} 1 :: ?)$ in ...	substitution steps
→*	$\sigma := ?, \sigma' := ?$	▷ let $n = \pi_2(su_\varepsilon^\sigma)(\langle \text{Int}, \sigma^{\text{Int}} \rangle 1 :: ?)$ in $n + 1$	argument is sealed by σ
→*	$\sigma := ?, \sigma' := ?$	▷ let $n = \varepsilon_{\text{Int}} 1 :: ?$ in $n + 1$	unsealing eliminates σ
→*	$\sigma := ?, \sigma' := ?$	▷ $\varepsilon_{\text{Int}} 2 :: ?$	

If we slightly modify the previous λ_{seal} program by $\nu x.\nu y.\text{let } \{n\}_y = \{1\}_x \text{ in } n + 1$, then unsealing fails with **unseal_error** because it uses a different seal to unseal than the one used to seal. The

embedding of this λ_{seal} term in GSF is very similar to the previous one; the only difference is that, now, u is bound to y . The following reduction trace illustrates where the embedding of the λ_{seal} term fails. Note that the resulting value of $\pi_2(su_{\varepsilon}^{\sigma'})$ is $\langle ? \rightarrow \sigma'^?, ? \rightarrow ? \rangle (\lambda x : ?. \langle \sigma'^?, \sigma'^? \rangle x :: \sigma') :: ? \rightarrow ?$, where σ' is a type name and $\sigma'^?$ is an evidence type. Then, the sealed value $\langle \text{Int}, \sigma^{\text{Int}} \rangle 1 :: ?$ is substituted in the body of the function, failing in the consistent transitivity $\langle \text{Int}, \sigma^{\text{Int}} \rangle \S \langle \sigma'^?, \sigma'^? \rangle$.

	▷ let $x = su_{\varepsilon}$ in let $y = su_{\varepsilon}$ in ...	initial program
\mapsto^*	$\sigma := ?, \sigma' := ?$ ▷ let $u = su_{\varepsilon}^{\sigma'}$ in ... let $s = su_{\varepsilon}^{\sigma}$ in ...	σ and σ' are generated
\mapsto^*	$\sigma := ?, \sigma' := ?$ ▷ let $z = \pi_1(su_{\varepsilon}^{\sigma})(\varepsilon_{\text{Int}} 1 :: ?)$ in ...	substitution steps
\mapsto^*	$\sigma := ?, \sigma' := ?$ ▷ let $n = \pi_2(su_{\varepsilon}^{\sigma'}) (\langle \text{Int}, \sigma^{\text{Int}} \rangle 1 :: ?)$ in $n + 1$	argument is sealed by σ
\mapsto^*	error	error unsealing by σ'

4.5 Semantic Preservation of the λ_{seal} Embedding in GSF

We now prove that the embedding of λ_{seal} into GSF is correct, namely that a λ_{seal} term and its translation to GSF behave similarly: either they both terminate to a value, both diverge, or both yield an error. It is worth pointing out that the semantic preservation theorem below only accounts for what we call *valid* λ_{seal} terms, *i.e.* terms that do not produce runtime type errors related to sealing, *i.e.* **seal_type_error**. We come back to this point at the end of this section. We write $t \Downarrow$ or $t \Downarrow v \parallel \mu$ if $t \parallel \cdot \mapsto^* v \parallel \mu$, for some v and μ . We write $t \Uparrow$ if t diverges, and $t \Downarrow \mathbf{error}$ if $t \parallel \cdot \mapsto^* \mathbf{error}$, where **error** \triangleq **type_error** or **unseal_error**. As before, we write $t \Downarrow$ if $\vdash t \rightsquigarrow t_{\varepsilon} : ?$ and $\cdot \triangleright t_{\varepsilon} \mapsto^* \Xi \triangleright v$, for some v and Ξ .

Theorem 4.1 (Embedding of λ_{seal}) *Let t be a valid closed λ_{seal} term.*

- a. $\vdash [t] : ?$
- b. $t \Downarrow$ implies $[t] \Downarrow$
- c. $t \Uparrow$ implies $[t] \Uparrow$
- d. $t \Downarrow \mathbf{error}$ implies $[t] \Downarrow \mathbf{error}$

To prove Theorem 4.1, we use a simulation relation \approx between λ_{seal} and GSF_{ε} , defined in Figure 4.3. The simulation relation $\mu, \Xi; \Gamma \vdash t \approx t_{\varepsilon} : ?$ uses a set of allocated seals μ by the reduction of the λ_{seal} term t . The GSF_{ε} term t_{ε} typechecks in the typing environment Γ where all variables have type unknown, and it typechecks and it is evaluated in the store Ξ with all its type names instantiated, also, to the unknown type. In all the rules of the simulation, we implicitly assume that μ and Ξ are synchronized, *i.e.* if $\sigma \in \mu$ then $\sigma := ? \in \Xi$. Rules whose names begin with (TR) relate a λ_{seal} term and its translation in GSF_{ε} , *i.e.* embedding first the λ_{seal} term into GSF, and then translating the resulting GSF term to a GSF_{ε} term. For instances, Rule (TRb) relates the λ_{seal} value $\mathbf{1}$ with the GSF_{ε} value $\varepsilon_{\text{Int}} 1 :: ?$. Note that Rule (TRp) uses metavariable D to denote the possible types of GSF_{ε} raw values (u), obtained by the embedding: either a base type B , an unknown function type $? \rightarrow ?$, or a pair of raw values $D \times D$. Rule (TRsG) relates

$$\begin{array}{c}
\text{(TRx)} \frac{x : ? \in \Gamma}{\mu; \Xi; \Gamma \vdash x \approx x : ?} \quad \text{(TRb)} \frac{ty(b) = B}{\mu; \Xi; \Gamma \vdash b \approx_{\mathcal{E}B} b :: ? : ?} \quad \text{(TRs)} \frac{\sigma \in \mu \quad \sigma := ? \in \Xi}{\mu; \Xi; \Gamma \vdash \sigma \approx su_{\mathcal{E}}^{\sigma} : ?} \\
\text{(TRp)} \frac{\mu; \Xi; \Gamma \vdash v_1 \approx_{\mathcal{E}D_1} u_1 :: ? : ? \quad \mu; \Xi; \Gamma \vdash v_2 \approx_{\mathcal{E}D_2} u_2 :: ? : ?}{\mu; \Xi; \Gamma \vdash \langle v_1, v_2 \rangle \approx_{\mathcal{E}D_1 \times D_2} \langle u_1, u_2 \rangle :: ? : ?} \\
\text{(TR}\lambda\text{)} \frac{\mu; \Xi; \Gamma, x : ? \vdash t_1 \approx t_2 : ?}{\mu; \Xi; \Gamma \vdash (\lambda x. t_1) \approx_{\mathcal{E}\lambda} (\lambda x. t_2) :: ? : ?} \quad \text{(TRpt)} \frac{\mu; \Xi; \Gamma \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi; \Gamma \vdash t_2 \approx t'_2 : ?}{\mu; \Xi; \Gamma \vdash \langle t_1, t_2 \rangle \approx_{\mathcal{E}\lambda} \langle t'_1, t'_2 \rangle :: ? : ?} \\
\text{(R?)} \frac{\mu; \Xi; \Gamma \vdash t \approx t' : ?}{\mu; \Xi; \Gamma \vdash t \approx_{\mathcal{E}\lambda} t' :: ? : ?} \quad \text{(Rapp)} \frac{\mu; \Xi; \Gamma \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma \vdash v_2 \approx v'_2 : ?}{\mu; \Xi; \Gamma \vdash v_1 v_2 \approx (\mathcal{E}\lambda) v'_1 v'_2 :: ? \rightarrow ?} \\
\text{(TRappL)} \frac{\mu; \Xi; \Gamma \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi; \Gamma \vdash t_2 \approx t'_2 : ?}{\mu; \Xi; \Gamma \vdash t_1 t_2 \approx \text{let } x = t'_1 \text{ in let } y = t'_2 \text{ in } (\mathcal{E}\lambda) y : ?} \\
\text{(RappR)} \frac{\mu; \Xi; \Gamma \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma \vdash t_2 \approx t'_2 : ?}{\mu; \Xi; \Gamma \vdash v_1 t_2 \approx \text{let } y = t'_2 \text{ in } (\mathcal{E}\lambda) v'_1 y : ?} \\
\text{(TRpi)} \frac{\mu; \Xi; \Gamma \vdash t \approx t' : ?}{\mu; \Xi; \Gamma \vdash \pi_i(t) \approx \pi_i(\mathcal{E}\lambda) t' :: ? \times ?} : ? \quad \text{(TRsG)} \frac{\mu; \Xi; \Gamma, x : ? \vdash t \approx t' : ?}{\mu; \Xi; \Gamma \vdash \nu x. t \approx \text{let } x = su_{\mathcal{E}} \text{ in } t' : ?} \\
\text{(Rsed1)} \frac{\mu; \Xi; \Gamma \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma \vdash v_2 \approx v'_2 : ?}{\mu; \Xi; \Gamma \vdash \{v_1\}_{v_2} \approx (\mathcal{E}\lambda) \pi_1(\mathcal{E}\lambda) v'_2 :: ? \times ?} : ? \rightarrow ? v'_1 : ? \\
\text{(TRsed1L)} \frac{\mu; \Xi; \Gamma \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi; \Gamma \vdash t_2 \approx t'_2 : ?}{\mu; \Xi; \Gamma \vdash \{t_1\}_{t_2} \approx \text{let } x = t'_1 \text{ in let } y = t'_2 \text{ in } (\mathcal{E}\lambda) \pi_1(\mathcal{E}\lambda) y :: ? \times ?} : ? \rightarrow ? x : ? \\
\text{(Rsed1R)} \frac{\mu; \Xi; \Gamma \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma \vdash t_2 \approx t'_2 : ?}{\mu; \Xi; \Gamma \vdash \{v_1\}_{t_2} \approx \text{let } y = t'_2 \text{ in } (\mathcal{E}\lambda) \pi_1(\mathcal{E}\lambda) y :: ? \times ?} : ? \rightarrow ? v'_1 : ? \\
\text{(Rsed2)} \frac{\mu; \Xi; \Gamma \vdash v \approx \langle E_1, E_2 \rangle u :: ? : ? \quad \sigma \in \mu \quad \sigma := ? \in \Xi}{\mu; \Xi; \Gamma \vdash \{v\}_{\sigma} \approx \langle E_1, \sigma^{E_2} \rangle u :: ? : ?} \\
\text{(Runs)} \frac{\mu; \Xi; \Gamma \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma \vdash v_2 \approx v'_2 : ? \quad \mu; \Xi; \Gamma, z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi; \Gamma \vdash \text{let } \{z\}_{v_1} = v_2 \text{ in } t_3 \approx \text{let } z = (\mathcal{E}\lambda) \pi_2(\mathcal{E}\lambda) v'_1 :: ? \times ?} : ? \rightarrow ? v'_2 \text{ in } t'_3 : ? \\
\text{(TRunsL)} \frac{\mu; \Xi; \Gamma \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi; \Gamma \vdash t_2 \approx t'_2 : ? \quad \mu; \Xi; \Gamma, z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi; \Gamma \vdash \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 \approx \text{let } x = t'_1 \text{ in let } y = t'_2 \text{ in} \\ \text{let } z = (\mathcal{E}\lambda) \pi_2(\mathcal{E}\lambda) x :: ? \times ?} : ? \rightarrow ? y \text{ in } t'_3 : ? \\
\text{(RunsR)} \frac{\mu; \Xi; \Gamma \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma \vdash t_2 \approx t'_2 : ? \quad \mu; \Xi; \Gamma, z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi; \Gamma \vdash \text{let } \{z\}_{v_1} = t_2 \text{ in } t_3 \approx \text{let } y = t'_2 \text{ in let } z = (\mathcal{E}\lambda) \pi_2(\mathcal{E}\lambda) v'_1 :: ? \times ?} : ? \rightarrow ? y \text{ in } t'_3 : ?
\end{array}$$

Figure 4.3: Simulation Relation Between λ_{seal} and $\text{GSF}_{\mathcal{E}}$ terms

the seal generation term $\nu x.t$ with the $\text{GSF}\varepsilon$ term that let-binds the variable x to the term su_ε to be substituted in t' ; it requires that the bodies of the seal generation and let-binding be related. The remaining rules, whose names begin with (R), help us keep terms related as they reduce. One of the most important rules is (Rsed2), which relates a λ_{seal} sealed value with a $\text{GSF}\varepsilon$ value that has sealing evidence, where σ is a type name and σ^{E_2} is an evidence type. Rule (Rsed1) relates a sealed value $\{v_1\}_{v_2}$ with a $\text{GSF}\varepsilon$ term that takes the first component of v'_2 (expected to be a su_ε^σ value related to the seal v_2), and applies it to v'_1 related to v_1 . Dually, Rule (Runs) relates a term for unsealing with a $\text{GSF}\varepsilon$ term that takes the second component of v'_1 (expected to be a su_ε^σ value related to the seal v_1), and applies it to v'_2 related to v_2 (expected to be a sealed value). Also, the rule requires the bodies t_3 and t_3 to be related.

We first establish a number of useful lemmas. First, all $\text{GSF}\varepsilon$ terms that are in the relation have type unknown, simulating the fact that they are related to untyped λ_{seal} terms.

Lemma 4.2 *If $\mu; \Xi; \Gamma \vdash t \approx t_\varepsilon : ?$ then $\Xi; \Gamma \vdash t_\varepsilon : ?$.*

Also, the relation \approx guarantees that if we have a λ_{seal} value related to a $\text{GSF}\varepsilon$ term, then the latter reduces to a related value.

Lemma 4.3 *If $\mu; \Xi \vdash v \approx t_\varepsilon : ?$, then there exists v_ε s.t. $\Xi \triangleright t_\varepsilon \mapsto^* \Xi \triangleright v_\varepsilon$, and $\mu; \Xi \vdash v \approx v_\varepsilon : ?$.*

For example, we know by Rule (Rsed1L) that $(\mu = \{\sigma\} \text{ and } \Xi = \{\sigma := \text{?}\})$

$$\mu; \Xi \vdash \{1\}_\sigma \approx \text{let } x = \varepsilon_{\text{Int}} 1 :: ? \text{ in let } y = su_\varepsilon^\sigma \text{ in } (\varepsilon_{? \rightarrow ?} \pi_1(\varepsilon_{? \times ?} y :: ? \times ?) :: ? \rightarrow ?) x : ?$$

Thus, we know that the $\text{GSF}\varepsilon$ term reduces to a value related to $\{1\}_\sigma$, in this case, $\langle \text{Int}, \sigma^{\text{Int}} \rangle 1 :: ?$.

Lemma 4.4 establishes substituting related values in related terms yields related terms.

Lemma 4.4 *If $\mu; \Xi; \Gamma, x : ? \vdash t \approx t_\varepsilon : ?$ and $\mu; \Xi; \Gamma \vdash v \approx v_\varepsilon : ?$, then $\mu; \Xi; \Gamma \vdash t[v/x] \approx t_\varepsilon[v_\varepsilon/x] : ?$.*

Lemma 4.5 shows that the relation \approx simulates both the notions of reduction \longrightarrow and \longrightarrow , and the reduction relations \mapsto and \mapsto , including error cases. Note that a single step of reduction in λ_{seal} can be simulated by several reduction steps in $\text{GSF}\varepsilon$, hence the use of \mapsto^* in the conclusions of the lemma cases. For example, we have $\mu; \Xi \vdash \pi_1(\langle 1, 2 \rangle) \approx \pi_1(\varepsilon_{? \times ?}(\varepsilon_{\text{Int} \times \text{Int}} \langle 1, 2 \rangle :: ?) :: ? \times ?) : ?$, and the $\text{GSF}\varepsilon$ term needs to reduce inside the frame $\pi_1(\square)$ before eliminating the projection like the λ_{seal} term.

Lemma 4.5 *Suppose that t is a term of λ_{seal} , t_ε is a term from $\text{GSF}\varepsilon$ and $\mu; \Xi \vdash t \approx t_\varepsilon : ?$.*

- a. *If $t \parallel \mu \longrightarrow t' \parallel \mu'$, then there exists t'_ε s.t. $\Xi \triangleright t_\varepsilon \mapsto^* \Xi' \triangleright t'_\varepsilon$ and $\mu'; \Xi' \vdash t' \approx t'_\varepsilon : ?$*
- b. *If $t \parallel \mu \longrightarrow \text{error}$, then $\Xi \triangleright t_\varepsilon \mapsto^* \text{error}$*
- c. *If $t \parallel \mu \mapsto t' \parallel \mu'$, then there exists t'_ε s.t. $\Xi \triangleright t_\varepsilon \mapsto^* \Xi' \triangleright t'_\varepsilon$ and $\mu'; \Xi' \vdash t' \approx t'_\varepsilon : ?$*
- d. *If $t \parallel \mu \mapsto \text{error}$, then $\Xi \triangleright t_\varepsilon \mapsto^* \text{error}$*

PROOF. The proof is by induction on $\mu; \Xi \vdash t \approx t_\varepsilon : ?$ and by analysis of the different cases.

Case (a). Most of the cases use Lemma 4.3, Lemma 4.4 and the consistent transitivity relation.

Case (b). Most of the cases use Lemma 4.3 and the consistent transitivity relation.

Case (c.). The proof follows by cases analysis on $t \parallel \mu \mapsto t' \parallel \mu'$ and from Case (a.).

Case (d.). The proof follows by cases analysis on $t \parallel \mu \mapsto \mathbf{error}$ and from Case (b.).

□

The main property of the relation \approx is that related terms behave similarly:

Lemma 4.6 *If $\vdash t \approx t_\varepsilon : ?$ then*

- $t \Downarrow v \parallel \mu$ implies $\cdot \triangleright t_\varepsilon \mapsto^* \Xi \triangleright v_\varepsilon$, where $\mu; \Xi \vdash v \approx v_\varepsilon : ?$.
- $t \Uparrow$ implies t_ε diverges.
- $t \Downarrow \mathbf{error}$ implies $\cdot \triangleright t_\varepsilon \mapsto^* \mathbf{error}$.

PROOF. The proof is by case analysis on the reduction of t .

- Suppose $t \Downarrow v \parallel \mu$. Then $\cdot \triangleright t_\varepsilon \mapsto^* \Xi \triangleright v_\varepsilon$ and $\mu; \Xi \vdash v \approx v_\varepsilon : ?$ by Lemmas 4.3 and 4.5(c.).
- Suppose $t \Uparrow$. Then t_ε diverges by Lemma 4.5(c.).
- Suppose $t \Downarrow \mathbf{error}$, then $\cdot \triangleright t_\varepsilon \mapsto^* \mathbf{error}$ by Lemma 4.5(c. and d.).

□

Finally, a λ_{seal} term and its embedding into GSF_ε are related.

Lemma 4.7 *If $\vdash [t] \rightsquigarrow t_\varepsilon : ?$, then $\vdash t \approx t_\varepsilon : ?$.*

Semantics preservation (Theorem 4.1) follows from Lemma 4.6 and Lemma 4.7.

Leaking the Encoding. As mentioned earlier, the semantic preservation result does not account for λ_{seal} terms that can raise runtime seal type errors, **seal.type.error**. The reason is that, without further caution, the encoding of seals as pairs of functions could be abused. For instance, the term $\text{let } \{y\}_{\langle \lambda x.x, \lambda x.x \rangle} = 1 \text{ in } y$ raises a **seal.type.error** in λ_{seal} , because the expression that is supposed to produce a seal produces a pair of functions. Nevertheless, the embedding of this term in GSF reduces to 1. To properly deal with such cases—and therefore obtain a semantic preservation statement with equivalences instead of implications—would require introducing a primitive way in GSF of distinguishing “proper seals” produced by the translation from standard pairs of functions. A direct solution would be to exploit the data abstraction capabilities of System F, and hence GSF . Of course, in a statically-typed version of λ_{seal} (Pierce and Sumii, 2000), this problem is sidestepped because a **seal.type.error** can never occur at runtime. It would be desirable to have made a complete embedding of λ_{seal} , but we focus more on studying the dynamic sealing mechanism of λ_{seal} to protect abstract data and its relation to gradual parametricity.

Embedding of the Dynamically-Typed Language. Finally, a direct consequence of the semantics preservation theorem is that the embedding of λ_{dyn} is also correct; in fact the embedding result holds as stated by Siek et al. (2015a)(Theorem 2), with equivalences instead of implications:

Corollary 4.8 (Embedding of λ_{dyn}) *Let t be a closed λ_{dyn} term.*

- a. $\vdash [t] : ?$
- b. $t \Downarrow$ if and only if $[t] \Downarrow$
- c. $t \Uparrow$ if and only if $[t] \Uparrow$

This result follows from Theorem 4.1 combined with the fact that a `seal_type_error` simply cannot occur in λ_{dyn} , which has no sealing-related terms.

4.6 Related Work

The embedding of a cryptographic lambda calculus in GSF directly relates to unpublished work by Siek and Wadler (2016), which studies the connection between a polymorphic cast calculus (Ahmed et al., 2011) and a cryptographic lambda calculus based on that of Pierce and Sumii (2000). Their approach is similar to our own, though the technicalities differ: their translation targets a cast calculus, more akin to $\text{GSF}\varepsilon$, while we define the embedding directly at the level of GSF. Therefore our result is the first to relate the dynamic end of a gradual parametric *source* language and dynamic sealing. Also, our embedding is inspired by the type $Univ = \exists Y. \forall X. (X \rightarrow Y) \times (Y \rightarrow X)$ put forth by Devriese et al. (2018) in their analysis of full abstraction mentioned above. The type $Univ$ can be interpreted as stating the existence of a universal type Y , *i.e.* a type that all other types can be embedded into and extracted from. As they show by a parametricity argument, $Univ$ cannot be inhabited in System F. In GSF, however, the unknown type $?$ plays this role of a universal type. Interestingly, Siek and Wadler (2016) discuss an alternative embedding that resembles ours, although they discard it as it does not align well with their treatment of blame.

4.7 Conclusion

This Chapter studied the dynamic end of the gradual typing spectrum supported by GSF, and showed that beyond a standard dynamically-typed language, GSF can faithfully embed a language with dynamic sealing primitives. The embedding is built using a general seal/unseal generator, which is expressed as a GSF term. The proposed term is a polymorphic pair of functions, of type $\forall X. (X \rightarrow ?) \times (? \rightarrow X)$. The first component of the pair serves a sealing function, producing an opaque value as a result, and the second component serves as the corresponding unsealing function, returning the original unsealed value. As we saw, the fact that a seal α is global once created, being able to continue existing when the type variable X for which it was created goes out of scope is what enables GSF to embed the cryptographic lambda calculus. However, at the same time, global seals have been shown to break equivalences that hold in System F (Devriese et al., 2018). In Chapter 6, we will return to this topic, proposing a new approach for gradual parametricity that avoids the global type name store, but contrary to GSF, it is not able to embed an untyped language with dynamic sealing primitives.

The next chapter presents the extension of GSF with existential types, which are the foundation of data abstraction and information hiding.

Chapter 5

Gradual Existential Types in GSF

Existential types are the foundation of data abstraction and information hiding: concrete representations of abstract data types are elements of existential types (Mitchell and Plotkin, 1988; Pierce, 2002). It is well known that existential types can be encoded in terms of universal types (Pierce, 2002). However, several polymorphic languages (Ahmed, 2006; Ahmed et al., 2009a; Neis et al., 2009) include both universal and existential types primitively, instead of relying on the encoding. The reason is that proving certain properties, such as representation independence results, is much simpler with direct support for existential types.

Although some efforts have already been developed to protect data abstraction in a dynamically-typed language (Morris, 1973; Abadi et al., 1995; Rossberg, 2003; Sumii and Pierce, 2004; Wadler, 2017), prior work on gradual parametric polymorphism leaves the treatment of existential types as future work (Ahmed et al., 2017). In this chapter, we present an extension of GSF with existential types, dubbed GSF^\exists . We first briefly review existential types (§5.1) and why a direct treatment is preferable to an encoding (§5.2). We then informally introduce gradual existential types in action (§5.3) before formally developing GSF^\exists (§5.4). Finally, we discuss the metatheory of GSF^\exists (§5.5).

5.1 Existential Types in a Nutshell

An *abstract data type* (ADT for short) guarantees that a client can neither guess nor depend on its implementation (Reynolds, 1983; Mitchell and Plotkin, 1988). Formally, an ADT consists of a type name A , a concrete representation type T , implementations of some operations for creating, querying and manipulating values of type T , and an abstraction boundary enclosing the representation and operations (Pierce, 2002). Thus, an ADT provides a public name to a type but hides its representation. The *representation independence* property for an ADT establishes that we can change its representation without affecting clients. This property is a particularly useful application of relational parametricity (Reynolds, 1983); we can show that two different implementations of an ADT are contextually equivalent so long as there exists a relation between their concrete type representations that is preserved by their operations.

Data abstraction is formalized by extending System F with existential types, of the form $\exists X.T$. Elements of an existential type are usually called packages, written $\text{pack}\langle T', t \rangle$ as $\exists X.T$, where T' is the hidden representation type and the term component t has type $T[T'/X]$. The existential elimi-

nation construct $\text{unpack}\langle X, x \rangle = t_1$ in t_2 allows the components of the package to be accessed by a client, keeping the actual representation type hidden. Packages with different hidden representation types can inhabit the same existential type. Thus, we can implement an ADT in different ways, creating different existential packages.

For instance, consider a semaphore ADT with three operations: *bit* to create a semaphore, *flip* to produce a semaphore in the inverted state, and *read* to consult the state of the semaphore, as a `Bool`. We can encode such an ADT as an existential type with a triple:

$$\text{Sem} \equiv \exists X. X \times (X \rightarrow X) \times (X \rightarrow \text{Bool})$$

Alternatively, for readability, we can use a hypothetical record syntax:

$$\text{Sem} \equiv \exists X. \{ \text{bit} : X, \text{flip} : X \rightarrow X, \text{read} : X \rightarrow \text{Bool} \}$$

Below are two equivalent implementations of this *Sem* ADT:

$$\begin{aligned} s_1 &\equiv \text{pack}\langle \text{Bool}, v_1 \rangle \text{ as Sem} && \text{where } v_1 \equiv \{ \text{bit} = \text{true}, \text{flip} = (\lambda x : \text{Bool}. \neg x), \text{read} = (\lambda x : \text{Bool}. x) \} \\ s_2 &\equiv \text{pack}\langle \text{Int}, v_2 \rangle \text{ as Sem} && \text{where } v_2 \equiv \{ \text{bit} = 1, \text{flip} = (\lambda x : \text{Int}. 1 - x), \text{read} = (\lambda x : \text{Int}. 0 < x) \} \end{aligned}$$

In the first implementation, the concrete representation type is `Bool`, and in the second it is `Int`. The representation and operations of the *Sem* ADT are abstract to a client, in the sense that the representation of *bit* is hidden, and it can only be manipulated and queried by the operations *flip* and *read*. For instance, if we have the expression $\text{unpack}\langle X, x \rangle = s$ in t , where s is an implementation of *Sem*, we can do $(x.\text{read} (x.\text{flip} x.\text{bit}))$ in the expression t , but $(x.\text{read} (x.\text{flip} \text{true}))$ or $x.\text{bit} == \text{true}$ are invalid programs that do not typecheck. Note that untyped versions of these programs would run normally with s_1 , but they would crash with s_2 .

5.2 Existential Types: Primitive or Encoded?

Existential types are closely connected with universal types, and in fact they can simply be *encoded* in terms of universal types, using the following encoding (Harper, 2012):

$$\begin{aligned} \exists X. T &\equiv \forall Y. (\forall X. T \rightarrow Y) \rightarrow Y \\ \text{pack}\langle T', t \rangle \text{ as } \exists X. T &\equiv \Lambda Y. \lambda f : (\forall X. T \rightarrow Y). f [T'] t \\ \text{unpack}\langle X, x \rangle = t_1 \text{ in } t_2 &\equiv t_1 [T_2] (\Lambda X. \lambda x : T. t_2) \text{ where } \Delta; \Gamma \vdash t_1 : \exists X. T \text{ and } \Delta, X; \Gamma, x : T \vdash t_2 : T_2 \end{aligned}$$

The intuition behind this encoding is that an existential type is viewed as a universal type taking the overall result type Y , followed by a polymorphic function representing the client with result type Y , and yielding a value of type Y as result. A package is a polymorphic function taking the client as argument, and unpacking corresponds to applying this polymorphic function.

Therefore, to study gradual existential types in GSF, one could simply adopt this encoding. However, if we want to reason about interesting properties such as representation independence and free theorems, it is preferable to give meaning to existential types directly.

The benefit of a direct treatment of existential types can already be appreciated in the fully-static setting, with the simple examples of packages s_1 and s_2 above. Suppose we want to show that s_1 and s_2 are contextually equivalent, *i.e.* indistinguishable by any context. To prove this equivalence, it is sufficient to show that the packages are related according to a parametricity logical relation that is sound with respect to contextual equivalence (Reynolds, 1983). Using the direct interpretation of existential types, such a proof is considerably easier and more intuitive than using their universal encodings.¹

The additional complexity of reasoning about existential types via their universal encoding hardly scales to more involved examples. For instance, Ahmed et al. (2009a) prove challenging cases of equivalences in the presence of abstract data types and mutable references, where the encoding would have been a liability; hence their choice of supporting existential types directly. Considering that the GSF logical relation also involves a number of technicalities (evidence, worlds, etc.), providing direct support for existentials is all the more appealing.

5.3 Gradual Existential Types in GSF[∇]

In this section, we show some illustrative examples of gradual existential types in action, highlighting their benefits and expected properties when type imprecision is involved. In particular, we want to dynamically preserve the information hiding property presumed for abstract data types. This property is conserved thanks to the dynamic sealing generation mechanism, similar to the parametricity preservation in gradual universal types.

Typed-Untyped Interoperability. Gradual existential types allow programmers to embed an untyped implementation of a library as a static ADT, by picking the unknown type as the hidden representation type. For instance, if v_3 is an untyped record, then s_3 below is a gradually well-typed implementation of the *Sem* ADT. The translation $[\cdot]$ embeds untyped terms in the gradual language, basically by introducing $?$ on all binders and constants (Siek and Taha, 2006).

$$\begin{aligned} &\text{let } v_3 = \{bit = 1, flip = (\lambda x.1 - x), read = (\lambda x.0 < x)\} \text{ in} \\ &\text{let } s_3 = \text{pack}(?, [v_3]) \text{ as } Sem \text{ in } C[s_3] \\ &\text{where } C \equiv \text{unpack}\langle X, x \rangle = \square \text{ in } (x.read (x.flip x.bit)) \end{aligned}$$

The package s_3 is essentially a version of the package s_2 where types have been erased (replaced with the unknown type). As illustrated later (§5.5.4), one can prove in GSF[∇] that s_3 is contextually equivalent to s_2 (and hence to s_1 as well), using a direct interpretation of gradual existential types. The static client or context C , given a package implementation of the *Sem* ADT, changes the state of the semaphore and then reads the state. The whole example runs without error, producing `false` as the final result.

Of course, we could have associated a package implementation that does not respect the ADT signature. For instance, we define v'_3 as a variant of v_3 , where *flip* has type $? \rightarrow \text{Bool}$. We obtain the package s'_3 , which is still gradually well-typed. However, using the package with client C results in a runtime type error. The runtime error happens when the \neg operator is applied to $x.bit$, because \neg expects a `Bool` argument, but dynamically *bit* is an `Int`.

¹ In the appendix, we provide sketches of these two proof techniques in System F.

```

let  $v'_3 = \{bit = 1, flip = (\lambda x. \neg x), read = (\lambda x. 0 < x)\}$  in
let  $s'_3 = \text{pack}\langle ?, [v'_3] \rangle$  as Sem in  $C[s'_3]$ 

```

The dual case of typed/untyped interoperability is that of a static package being used in dynamic code. The following example defines the untyped function g , which take as arguments the function f and an expression x to be applied to f . The function g is applied to the typed components of the package s_2 , reducing the whole program without error to `true`.

```

let  $g = (\lambda f. \lambda x. f x)$  in  $\text{unpack}\langle X, x \rangle = s_2$  in  $((g x.read) x.bit)$ 

```

Taking the same example, but changing $x.bit$ to the expression `(1 :: ?)` yields a runtime error, because the function $x.read$ is expecting a sealed value, but instead it receives an unsealed `Int`.

```

let  $g = (\lambda f. \lambda x. f x)$  in  $\text{unpack}\langle X, x \rangle = s_2$  in  $((g x.read) (1 :: ?))$ 

```

Optimistic Type Checking. The following example shows how the optimistic gradual type checker accepts programs that run without errors, which would be rejected with a static type checker.

```

 $\text{unpack}\langle X, x \rangle = s_2$  in
let  $f = \lambda z. \text{if}(z)$  then  $(x.flip :: ?)$  else  $((\lambda x : \text{Int}. 1 - x) :: ?)$  in
let  $v'_2 = \{bit = x.bit, flip = f \text{ true}, read = x.read\}$  in
let  $s'_2 = \text{pack}\langle X, v'_2 \rangle$  as Sem in
 $\text{unpack}\langle Y, y \rangle = s'_2$  in  $(y.read (y.flip y.bit))$ 

```

The package s'_2 is essentially the same as s_2 —in fact they are equivalent. The function f receives a `Bool` argument to decide whether to return the (hidden) `flip` function from package s_2 , or a literal (not hidden) function. This program is gradually well-typed because of the ascriptions to the unknown type in the branches of the conditional. In contrast, a static type system would reject this program (without the `?` ascriptions in the conditional branches) because the then branch would have type $X \rightarrow X$, while the else branch would have type $\text{Int} \rightarrow \text{Int}$. The gradual program runs properly, yielding `false` as a result. Note that if the definition of `flip` in v'_2 would be `f false`, then a runtime error would be raised. The error would be produced during the evaluation of the definition of s'_2 because v'_2 ought to have type $X \times (X \rightarrow X) \times (X \rightarrow \text{Bool})$, but instead it would have type $X \times (\text{Int} \rightarrow \text{Int}) \times (X \rightarrow \text{Bool})$.

Internal vs. External Imprecision. Another point to take into account is the nature of the imprecision of a term of existential type. As discussed previously regarding universal types (§3.6.2), the imprecision for existential types can be either internal or external, and this has an impact on

runtime behavior. The following program is fully static except for the imprecise ascription of s_2 to the type $Sem_1 \equiv \exists X.X \times (X \rightarrow ?) \times (X \rightarrow \text{Bool})$. Observe that $Sem \sqsubseteq Sem_1$.

$$\text{unpack}\langle X, x \rangle = s_2 :: Sem_1 \text{ in } (x.\text{read } (x.\text{flip } (x.\text{flip } x.\text{bit})))$$

Here we are in the presence of an ascribed imprecision (*i.e.* external), preserving the GSF property that if we ascribe a static closed term to a less precise type, its behavior is preserved: this program runs without error, and evaluates to `true`. Indeed, we will later show that GSF^\exists satisfies the weak dynamic gradual guarantee DGG^{\leq} (§5.5).

Conversely, in the following example, the imprecision is now internal, due to the imprecise signature Sem_1 of the package.

$$\text{unpack}\langle X, x \rangle = \text{pack}\langle \text{Int}, v_2 \rangle \text{ as } Sem_1 \text{ in } (x.\text{flip } (x.\text{flip } x.\text{bit})) + 10$$

This program is accepted statically, but fails at runtime because according to type-driven sealing, it would otherwise reveal hidden information, namely, the fact that the supposedly-hidden representation type is `Int`. The function $x.\text{flip}$ has type $X \rightarrow ?$, which specifies that it has to be applied to a sealed value and could return another sealed value, or in this case, an `Int` value. The application $(x.\text{flip } x.\text{bit})$ has type $?$, and it is used as the argument of $x.\text{flip}$ again, optimistically treated as an abstract type. Then, the result of the second application of $x.\text{flip}$ is added to 10, being optimistic again with the result of the function $x.\text{flip}$, but this time at type `Int`. The program fails at runtime because of the attempt to use $x.\text{flip}$ with both types $X \rightarrow X$ and $X \rightarrow \text{Int}$. Note that if we allow both behaviors of the function $x.\text{flip}$, returning 11, then we would be revealing that the hidden representation type is `Int`. Thus, we admit at runtime the first application of $x.\text{flip}$, accepted only with the type $X \rightarrow \text{Int}$, but it fails in the second application because it receives an `Int` instead of a sealed value.

5.4 Semantics of GSF^\exists

In this section, we formally present the design and semantics of GSF^\exists , an extension of GSF with existential types that exhibits the behaviors illustrated above. First, we introduce the static language SF^\exists , which is the starting point to apply AGT. Actually, we only apply AGT to the new features in SF^\exists since the others have already been gradualized. Then, we focus on GSF^\exists , the static and dynamic semantics derived by AGT.

5.4.1 The Static Language SF^\exists

We derive GSF^\exists by applying AGT to SF extended with existential types, called SF^\exists (Figure 5.1). We extend SF statics with the rules (Tpack) and (Tunpack) for a package and its elimination form, which are standard. We augment the definition of type equality to deal with existential types, and use the schm_e function to extract the schema of an existential type.

$$\begin{aligned}
\alpha &\in \text{TYPE_NAME} & \Sigma &\in \text{TYPE_NAME} \stackrel{\text{fn}}{\rightarrow} \text{TYPE}, \Delta \subset \text{TYPE_VAR}, \Gamma \in \text{VAR} \stackrel{\text{fn}}{\rightarrow} \text{TYPE} \\
T & ::= \dots \mid \exists X.T & & \text{(types)} \\
t & ::= \dots \mid \text{pack}\langle T, t \rangle \text{ as } \exists X.T \mid \text{unpack}\langle X, x \rangle = t \text{ in } t & & \text{(terms)} \\
v & ::= \dots \mid \text{pack}\langle T, v \rangle \text{ as } \exists X.T & & \text{(values)}
\end{aligned}$$

$\boxed{\Sigma; \Delta; \Gamma \vdash t : T}$ **Well-typed terms**

$$\begin{aligned}
& \text{(Tpack)} \frac{\Sigma; \Delta; \Gamma \vdash t : T_1 \quad \Sigma; \Delta \vdash T_1 = T[T'/X] \quad \Sigma; \Delta \vdash T'}{\Sigma; \Delta; \Gamma \vdash \text{pack}\langle T', t \rangle \text{ as } \exists X.T : \exists X.T} \\
& \text{(Tunpack)} \frac{\Sigma; \Delta; \Gamma \vdash t_1 : T_1 \quad \Sigma; \Delta, X; \Gamma, x : \text{schm}_e(T_1) \vdash t_2 : T_2 \quad \Sigma; \Delta \vdash T_2}{\Sigma; \Delta; \Gamma \vdash \text{unpack}\langle X, x \rangle = t_1 \text{ in } t_2 : T_2}
\end{aligned}$$

$$\begin{aligned}
& \text{schm}_e : \text{TYPE} \rightarrow \text{TYPE} \\
& \text{schm}_e(\exists X.T) = T \\
& \text{schm}_e(T) \text{ undefined o/w}
\end{aligned}$$

$\boxed{\Sigma; \Delta \vdash T = T}$ **Type equality**

$$\frac{\Sigma; \Delta, X \vdash T_1 = T_2}{\Sigma; \Delta \vdash \exists X.T_1 = \exists X.T_2}$$

$\boxed{\Sigma \triangleright t \rightarrow \Sigma \triangleright t}$ **Notion of reduction**

$\Sigma \triangleright (\text{unpack}\langle X, x \rangle = \text{pack}\langle T', v \rangle \text{ as } \exists X.T \text{ in } t) \rightarrow \Sigma, \alpha := T' \triangleright t[\alpha/X][v/x]$ where $\alpha \notin \text{dom}(\Sigma)$

$\boxed{\Sigma \triangleright t \mapsto \Sigma \triangleright t}$ **Evaluation frames and reduction**

$f ::= \dots \mid \text{pack}\langle T, \square \rangle \text{ as } \exists X.T \mid \text{unpack}\langle X, x \rangle = \square \text{ in } t$ (term frames)

Figure 5.1: SF[∃]: Syntax, Static and Dynamic Semantics (extends Figure 3.1)

The dynamic semantics of the unpack constructor is very similar to the type application; also a fresh type name α is generated and bound to the representation type T' in the global type name store Σ . Then, we substitute α (instead of the representation type) and the term component, for the variables X and x in the body of the unpack. Like SF, SF[∃] is also type safe, and all well-typed terms are parametric. As usual, in SF[∃], two packages are related if their term components are related under some relations between their concrete type representations. We can define the interpretation of existential types in SF[∃] using the same auxiliary definitions for the logical relation of GSF. Some definitions, such as $\text{Atom}_\rho^\equiv[\exists X.G]$, become simpler because they do not need to deal with the evidence.

$$\begin{aligned}
\mathcal{V}_\rho[\exists X.G] = & \{ (W, \text{pack}\langle T_1, v_1 \rangle \text{ as } \exists X.\rho(G), \text{pack}\langle G_2, v_2 \rangle \text{ as } \exists X.\rho(G)) \in \text{Atom}_\rho^\equiv[\exists X.G] \mid \\
& \forall W' \succeq W, \alpha, \exists R \in \text{REL}_{W',j}[G_1, G_2]. (W' \boxtimes (\alpha, G_1, G_2, R), v_1, v_2) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[G] \}
\end{aligned}$$

5.4.2 GSF^\exists : Statics

We derive the statics of GSF^\exists following AGT. As in Section 3.2, we first define the syntax of gradual typing, and we give them meaning through the concretization function. Then, we lift the static semantics of the static language to gradual settings using the corresponding abstraction function, which forms a Galois connection. Being consistent with the above, we extend the syntactic category of gradual types $G \in \text{GTYPE}$ with existential types:

$$G ::= B \mid G \rightarrow G \mid \forall X.G \mid G \times G \mid X \mid \alpha \mid ? \mid \exists X.G$$

As usual, the unknown type represents any type, including existential types. We naturally extend the concretization function C and abstraction function A to existential types, preserving the Galois connection established earlier (Proposition 3.3):

$$C(\exists X.G) = \{\exists X.T \mid T \in C(G)\} \quad A(\{\overline{\exists X.T_i}\}) = \exists X.A(\{\overline{T_i}\})$$

We define in Figure 5.2 the inductive definition of type precision, which is equivalent to Definition 3.1 (Proposition 5.1). As a result, $\exists X.?$ denotes any existential type, being more precise than the unknown type and less precise than $\exists X.X \rightarrow X$.

With the meaning of gradual types, the GSF^\exists static semantics follow as usual with AGT. In this case, we need to define the gradual counterpart of the type equality predicate, whose lifting is type consistency. Following Definition B.96, we can find in Figure 5.2 an equivalent inductive characterization of type consistency (Proposition 5.2). Then, we lift functions using abstraction, concretization and Definition B.97. Our only new function in SF^\exists is schm_e , whose lifting schm_e^\sharp is defined in Figure 5.2 as expected.

The gradual typing rules of GSF^\exists (Figure 5.2) extend those of GSF . The new rules are obtained by replacing type predicates and functions with their corresponding consistent liftings in the static typing rules. Observe that Rule (Gpack) uses type consistency instead of type equality so that the implementation term can be of a type that is distinct from, but consistent with the package type (after substituting for the representation type). For example:

$$\text{pack}\langle \text{Bool}, v_1 \rangle \text{ as } \text{Sem}_3 \quad \text{where } \text{Sem}_3 \equiv \exists X.X \times (X \rightarrow X) \times (X \rightarrow ?)$$

Here, the type of v_1 is $\text{Bool} \times (\text{Bool} \rightarrow \text{Bool}) \times (\text{Bool} \rightarrow \text{Bool})$, which is more precise than $\text{Sem}_3 [\text{Bool}] \equiv \text{Bool} \times (\text{Bool} \rightarrow \text{Bool}) \times (\text{Bool} \rightarrow ?)$.

Rule (Gunpack) uses the consistent existential schema function schm_e^\sharp , which allows a term of unknown type to be optimistically treated as a package, and therefore unpacked.

5.4.3 GSF^\exists : Dynamics

We now turn to the dynamic semantics of GSF^\exists . As we did before, we give the dynamic semantics of GSF^\exists in terms of a more informative variant called GSF_e^\exists . In GSF_e^\exists , all values are ascribed, and ascriptions carry evidence.

$$\alpha \in \text{TYPERNAME} \quad \Xi \in \text{TYPERNAME} \xrightarrow{\text{fin}} \text{GTYPE}, \Delta \subset \text{TYPEVAR}, \Gamma \in \text{VAR} \xrightarrow{\text{fin}} \text{GTYPE}$$

$$G ::= \dots \mid \exists X.G \quad (\text{gradual types})$$

$$t ::= \dots \mid \text{pack}\langle G, t \rangle \text{ as } \exists X.G \mid \text{unpack}\langle X, x \rangle = t \text{ in } t \quad (\text{gradual terms})$$

$\boxed{\Xi; \Delta; \Gamma \vdash t : G}$ **Well-typed terms**

$$\text{(Gpack)} \frac{\Xi; \Delta; \Gamma \vdash t : G_1 \quad \Xi; \Delta \vdash G_1 \sim G[G'/X] \quad \Xi; \Delta \vdash G'}{\Xi; \Delta; \Gamma \vdash \text{pack}\langle G', t \rangle \text{ as } \exists X.G : \exists X.G}$$

$$\text{(Gunpack)} \frac{\Xi; \Delta; \Gamma \vdash t_1 : G_1 \quad \Xi; \Delta, X; \Gamma, x : \text{schm}_e^\sharp(G_1) \vdash t_2 : G_2 \quad \Xi; \Delta \vdash G_2}{\Xi; \Delta; \Gamma \vdash \text{unpack}\langle X, x \rangle = t_1 \text{ in } t_2 : G_2}$$

$$\text{schm}_e^\sharp : \text{GTYPE} \rightarrow \text{GTYPE}$$

$$\text{schm}_e^\sharp(\exists X.G) = G$$

$$\text{schm}_e^\sharp(?) = ?$$

$$\text{schm}_e^\sharp(G) \text{ undefined o/w}$$

$\boxed{\Xi; \Delta \vdash G \sim G}$ **Type consistency**

$$\frac{\Xi; \Delta, X \vdash G_1 \sim G_2}{\Xi; \Delta \vdash \exists X.G_1 \sim \exists X.G_2}$$

$\boxed{G \sqsubseteq G}$ **Type precision**

$$\frac{G_1 \sqsubseteq G_2}{\exists X.G_1 \sqsubseteq \exists X.G_2}$$

Figure 5.2: GSF[∃]: Syntax and Static Semantics (extends Figure 3.4)

Figure 5.3 presents the syntax, static and dynamics semantics of GSF_ε[∃]; essentially those of GSF_ε naturally extended with existential types. It is worth noting that we introduce the syntactic form $\text{packu}\langle G', v \rangle \text{ as } \exists X.G$ for raw existential values. The reduction rule (*Rpack*) reduces the term $\text{pack}\langle G', v \rangle \text{ as } \exists X.G$ to the value $\varepsilon_{\exists X.G} \text{packu}\langle G', v \rangle \text{ as } \exists X.G :: \exists X.G$, inserting the evidence $\varepsilon_{\exists X.G}$ (evidence of the reflexive judgment $\exists X.G \sim \exists X.G$), the ascription to $\exists X.G$ and changing the syntax of the package by packu . The reduction rule (*Runpack*) specifies the reduction of an unpack expression: we substitute a fresh type name α for X in the body of the unpack , as well as a (carefully ascribed) package implementation for x . In particular, this rule combines the evidence from the actual implementation term $\varepsilon_1 u :: G_1$ with the evidence of the package, substituting the representation type on the left G' and the fresh type name α on the right for the type variable X . Note that the evidence ε justifies that the static type of the package declared by the keyword “as” is consistent with $\exists X.G$. Thus $\varepsilon[\hat{G}', \hat{\alpha}]$ justifies that the static type after the substitution by G' is consistent with $G[\alpha/X]$. Formally, $\varepsilon[\hat{G}', \hat{\alpha}] = \langle p_1(\varepsilon) [\hat{G}'], p_2(\varepsilon) [\hat{\alpha}] \rangle$, where $\hat{G}' = \text{lift}_{\Xi'}(G')$ and $\hat{\alpha} = \text{lift}_{\Xi'}(\alpha)$. Consequently, the resulting evidence of $\varepsilon_1 ; \varepsilon[\hat{G}', \hat{\alpha}]$ justifies that the type of the implementation term is consistent with $G[\alpha/X]$. Failure to justify this judgment produces an error, specifying that the implementation term is not appropriate. This evidence plays a key role in making the implementation term abstract, *i.e.* ensuring information hiding.

$$\begin{aligned}
t &::= \dots \mid \text{pack}\langle G, t \rangle \text{ as } \exists X.G \mid \text{unpack}\langle X, x \rangle = t \text{ in } t && \text{(terms)} \\
u &::= \dots \mid \text{packu}\langle G', v \rangle \text{ as } \exists X.G && \text{(raw values)}
\end{aligned}$$

$\boxed{\Xi; \Delta; \Gamma \vdash s : G}$ **Well-typed terms**

$$\begin{aligned}
(\text{Epacku}) \frac{\Xi; \Delta; \Gamma \vdash v : G[G'/X] \quad \Xi; \Delta \vdash G'}{\Xi; \Delta; \Gamma \vdash \text{packu}\langle G', v \rangle \text{ as } \exists X.G : \exists X.G} & \quad (\text{Epack}) \frac{\Xi; \Delta; \Gamma \vdash t : G[G'/X] \quad \Xi; \Delta \vdash G'}{\Xi; \Delta; \Gamma \vdash \text{pack}\langle G', t \rangle \text{ as } \exists X.G : \exists X.G} \\
(\text{Eunpack}) \frac{\Xi; \Delta; \Gamma \vdash t_1 : \exists X.G_1 \quad \Xi; \Delta, X; \Gamma, x : G_1 \vdash t_2 : G_2 \quad \Xi; \Delta \vdash G_2}{\Xi; \Delta; \Gamma \vdash \text{unpack}\langle X, x \rangle = t_1 \text{ in } t_2 : G_2}
\end{aligned}$$

$\boxed{\Xi \triangleright t \longrightarrow \Xi \triangleright t \text{ or error}}$ **Notion of reduction**

$$(\text{Rpack}) \Xi \triangleright \text{pack}\langle G', v \rangle \text{ as } \exists X.G \longrightarrow \Xi \triangleright \varepsilon_{\exists X.G} \text{packu}\langle G', v \rangle \text{ as } \exists X.G :: \exists X.G$$

(Runpack)

$$\Xi \triangleright \text{unpack}\langle X, x \rangle = \varepsilon \text{packu}\langle G', \varepsilon_1 u :: G_1 \rangle :: \exists X.G \text{ in } t \longrightarrow \begin{cases} \Xi' \triangleright t[\hat{\alpha}/X][[(\varepsilon_1 \circ \varepsilon[\hat{G}', \hat{\alpha}])u :: G[\alpha/X]]/x] \\ \text{where } \Xi' \triangleq \Xi, \alpha := G' \text{ for some } \alpha \notin \text{dom}(\Xi) \\ \hat{G}' = \text{lift}_{\Xi'}(G') \text{ and } \hat{\alpha} = \text{lift}_{\Xi'}(\alpha) \\ \text{error} \quad \text{if not defined} \end{cases}$$

$\boxed{\Xi \triangleright t \mapsto \Xi \triangleright t \text{ or error}}$ **Evaluation frames and reduction**

$$f ::= \dots \mid \text{pack}\langle G, \square \rangle \mid \text{unpack}\langle X, x \rangle = \square \text{ in } t$$

Figure 5.3: $\text{GSF}_{\varepsilon}^{\exists}$: Syntax, Static and Dynamic Semantics (extends Figure 3.5)

To support the dynamic semantics for existential types, we need to extend the representation of evidence types E in $\text{GSF}_{\varepsilon}^{\exists}$, adding $\exists X.E$ for existential evidence types. Additionally, we extend the definitions of consistent transitivity naturally: consistent transitivity between evidences with existential types simply relies on the underlying schemes:

$$(\text{ex}) \frac{\langle E_1, E_2 \rangle \circ \langle E_3, E_4 \rangle = \langle E'_1, E'_2 \rangle}{\langle \exists X.E_1, \exists X.E_2 \rangle \circ \langle \exists X.E_3, \exists X.E_4 \rangle = \langle \exists X.E'_1, \exists X.E'_2 \rangle}$$

Illustration. We now return to the gradual semaphore implementation s_3^* , which is the translation of the term s_3 from GSF^{\exists} to $\text{GSF}_{\varepsilon}^{\exists}$. Remember that all base values in $\text{GSF}_{\varepsilon}^{\exists}$ are ascribed to their base types, but for simplicity below, we omit trivial evidences. The following reduction trace illustrates all the important aspects of reduction in $\text{GSF}_{\varepsilon}^{\exists}$:

		Initial evidence	$\text{unpack}\langle X, x \rangle = \varepsilon_{Sem} \text{packu}\langle ?, v_3^* \rangle :: Sem \text{ in } (x.read (x.flip x.bit))$
		Consistent transitivity	
$(Runpack, Rproj_i)$	\mapsto^*		$\langle\langle ? \rightarrow Bool, \alpha^? \rightarrow Bool \rangle (\lambda x. 0 < x) :: \alpha \rightarrow Bool \rangle$ $\langle\langle\langle ? \rightarrow Int, \alpha^? \rightarrow \alpha^{Int} \rangle (\lambda x. 1 - x) :: \alpha \rightarrow \alpha \rangle (\langle Int, \alpha^{Int} \rangle 1 :: \alpha) \rangle$
		Unsealing eliminates α	
$(Rapp)$	\mapsto		$\langle\langle ? \rightarrow Bool, \alpha^? \rightarrow Bool \rangle (\lambda x. 0 < x) :: \alpha \rightarrow Bool \rangle (\langle Int, \alpha^{Int} \rangle (1 - 1) :: \alpha)$
		The return is sealed	
$(Rop, Rasc)$	\mapsto^*		$\langle\langle ? \rightarrow Bool, \alpha^? \rightarrow Bool \rangle (\lambda x. 0 < x) :: \alpha \rightarrow Bool \rangle (\langle Int, \alpha^{Int} \rangle 0 :: \alpha)$
		Unsealing eliminates α	
$(Rapp)$	\mapsto		$\varepsilon_{Bool}(0 < 0) :: Bool$
$(Rop, Rasc)$	\mapsto^*		$\varepsilon_{Bool} \text{false} :: Bool$

In this example, the initial evidence of the package is fully static. We omit some steps in the reduction, but is crucially to show in the rule $(Runpack)$ how the evidence $\varepsilon_{Sem}[?, \alpha^?]$ is calculated:

$$\varepsilon_{Sem}[?, \alpha^?] \equiv \langle ? \times (? \rightarrow ?) \times (? \rightarrow Bool), \alpha^? \times (\alpha^? \rightarrow \alpha^?) \times (\alpha^? \rightarrow Bool) \rangle$$

After some application of the rule $(Rproj_i)$, the term component is protected by the type name α . The application step $(Rapp)$ then gives rise to unsealing evidence to interact with the implementation and sealing evidence to protect the implementation.

5.5 Properties of GSF^\exists

In this section, we summarize the main properties, statics and dynamics, concerning GSF^\exists . We cover the refined criteria for gradual typing and parametricity.

5.5.1 Static Properties

We can show that the GSF^\exists meet the same static properties as GSF .

Proposition 5.1 (GSF^\exists : Precision, inductively) *The inductive definition of type precision given in Figure 5.2 is equivalent to Definition 3.1.*

Proposition 5.2 (GSF^\exists : Consistency, inductively) *The inductive definition of type consistency given in Figure 5.2 is equivalent to Definition B.96.*

The type system of GSF^\exists is equivalent to the SF^\exists type system on fully-static terms (Proposition 5.3), where \vdash_S denote the typing judgment of SF^\exists .

Proposition 5.3 (GSF^\exists : Static equivalence for static terms) *Let t be a static term and G a static type ($G = T$). We have $\vdash_S t : T$ if and only if $\vdash t : T$.*

The static semantics of GSF^\exists satisfy the static gradual guarantee (Proposition 5.4), where type precision (Def. 3.1) extends naturally to *term* precision.

Proposition 5.4 (GSF^\exists : Static gradual guarantee) *Let t and t' be closed GSF^\exists terms such that $t \sqsubseteq t'$ and $\vdash t : G$. Then $\vdash t' : G'$ and $G \sqsubseteq G'$.*

5.5.2 Dynamic Gradual Guarantees

Not surprisingly, GSF^\exists does not satisfy the dynamic gradual guarantee (§3.6) with respect to precision \sqsubseteq for existential types. Let us return to the semaphore implementation s_1 . Note that $s_1 \sqsubseteq s_4$, where $s_4 = \text{pack}\langle \text{Bool}, v_1 \rangle$ as $\exists X.X \times (X \rightarrow X) \times (? \rightarrow \text{Bool})$. If we use these terms in the same context as follows, we will obtain that

$$\text{unpack}\langle X, x \rangle = s_1 \text{ in } (x.\text{read } (x.\text{flip } x.\text{bit})) \sqsubseteq \text{unpack}\langle X, x \rangle = s_4 \text{ in } (x.\text{read } (x.\text{flip } x.\text{bit}))$$

However, the term on the left reduces to 2, while the (less precise) term on the right produces a runtime error because of the attempt to apply the function *read* (in this case of type $? \rightarrow \text{Bool}$) to a sealed value. On the other hand, with a simple extension of strict precision to existential types, GSF^\exists does satisfy the weaker dynamic gradual guarantee DGG^{\leq} (Theorem 3.22). Figure 5.4 defines the strict type and term precision for both $\text{GSF}_\varepsilon^\exists$ and GSF^\exists .

5.5.3 Parametricity

We establish parametricity for GSF^\exists by proving parametricity for $\text{GSF}_\varepsilon^\exists$. We extend the step-indexed logical relation for GSF_ε (Figure 3.15), adding the interpretation of existential types. Usually, two packages are related if their term components are related under some conditions (Ahmed, 2006; Neis et al., 2009). But in gradual settings, the definition of $\mathcal{V}_\rho[\exists X.G]$ is more complex. We start with the classical interpretation of the existential types adapted to our previous logical relation (which is not adequate for the interpretation of gradual existential types):

$$\mathcal{V}_\rho[\exists X.G] = \{ (W, \varepsilon_1 \text{packu}\langle G_1, v_1 \rangle :: \exists X.\rho(G), \varepsilon_2 \text{packu}\langle G_2, v_2 \rangle :: \exists X.\rho(G)) \in \text{Atom}_\rho^{\bar{}}[\exists X.G] \mid \forall W' \succeq W, \alpha. \exists R \in \text{REL}_{W',j}[G_1, G_2]. (W' \boxtimes (\alpha, G_1, G_2, R), v_1, v_2) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[G] \}$$

Let us focus on some simple and not very interesting programs but useful to explain why this existential type interpretation is not suitable. For example, if we want to relate these two package, $\varepsilon_{\exists X.X} \text{packu}\langle \text{Int}, \varepsilon_{\text{Int}} 1 :: \text{Int} \rangle :: \exists X.X$ and $\varepsilon_{\exists X.X} \text{packu}\langle \text{Bool}, \varepsilon_{\text{Bool}} \text{true} :: \text{Bool} \rangle :: \exists X.X$, under the above definition, then we would have to prove that their component terms, $\varepsilon_{\text{Int}} 1 :: \text{Int}$ and $\varepsilon_{\text{Bool}} \text{true} :: \text{Bool}$, are related in $\mathcal{V}_{\rho[X \mapsto \alpha]}[X]$, which is not true. Keep in mind that for two terms to be related in our logical relation they must have the same type, and they are related in a type variable if they are related in the type variable substituted by its associated type name.

Taking the above into account, we change the logical interpretation of existential types slightly. It is worth pointing out that this definition is not enough to interpret existential types.

$G \leq G$ **Strict type precision**

$$\frac{G_1 \leq G_2}{\exists X.G_1 \leq \exists X.G_2}$$

$\Omega \vdash \Xi_1 \triangleright s : G \leq \Xi_2 \triangleright s : G$ **Strict term precision** (for conciseness, s ranges over both t and u)

$$\begin{aligned} (\leq \text{packu}_\varepsilon) & \frac{G'_1 \leq G'_2 \quad \Omega \vdash \Xi_1 \triangleright v_1 : G_1[G'_1/X] \leq \Xi_2 \triangleright v_2 : G_2[G'_2/X] \quad \exists X.G_1 \sqsubseteq \exists X.G_2}{\Omega \vdash \Xi_1 \triangleright \text{packu}\langle G'_1, v_1 \rangle \text{ as } \exists X.G_1 : \exists X.G_1 \leq \Xi_2 \triangleright \text{packu}\langle G'_2, v_2 \rangle \text{ as } \exists X.G_2 : \exists X.G_2} \\ (\leq \text{pack}_\varepsilon) & \frac{G'_1 \leq G'_2 \quad \Omega \vdash \Xi_1 \triangleright t_1 : G_1[G'_1/X] \leq \Xi_2 \triangleright t_2 : G_2[G'_2/X] \quad \exists X.G_1 \leq \exists X.G_2}{\Omega \vdash \Xi_1 \triangleright \text{pack}\langle G'_1, t_1 \rangle \text{ as } \exists X.G_1 : \exists X.G_1 \leq \Xi_2 \triangleright \text{pack}\langle G'_2, t_2 \rangle \text{ as } \exists X.G_2 : \exists X.G_2} \\ (\leq \text{unpack}_\varepsilon) & \frac{\Omega \vdash \Xi_1 \triangleright t_1 : \exists X.G_1 \leq \Xi_2 \triangleright t_2 : \exists X.G_2 \quad \Omega, x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright t'_1 : G'_1 \leq \Xi_2 \triangleright t'_2 : G'_2}{\Omega \vdash \Xi_1 \triangleright \text{unpack}\langle X, x \rangle = t_1 \text{ in } t'_1 : G'_1 \leq \Xi_2 \triangleright \text{unpack}\langle X, x \rangle = t_2 \text{ in } t'_2 : G'_2} \end{aligned}$$

$G \rightarrow G$ **Type matching**

$$\exists X.G \rightarrow \exists X.G \quad ? \rightarrow \exists X.?$$

$\Omega \vdash v : G \leq_v v : G$ **Strict value precision**

$$(\leq \text{packu}) \frac{G'_1 \leq G'_2 \quad \Omega \vdash v_1 : G'_1 \leq_v v_2 : G'_2 \quad \exists X.G_1 \sqsubseteq \exists X.G_2 \quad G'_1 \sqcap G_1[G'_1/X] \leq G'_2 \sqcap G_2[G'_2/X]}{\Omega \vdash \text{pack}\langle G'_1, v_1 \rangle \text{ as } \exists X.G_1 : \exists X.G_1 \leq_v \text{pack}\langle G'_2, v_2 \rangle \text{ as } \exists X.G_2 : \exists X.G_2}$$

$\Omega \vdash \Xi_1 \triangleright t : G \leq \Xi_2 \triangleright t : G$ **Strict term precision**

$$\begin{aligned} (\leq \text{pack}) & \frac{G'_1 \leq G'_2 \quad \Omega \vdash t_1 : G'_1 \leq t_2 : G'_2 \quad \exists X.G_1 \leq \exists X.G_2 \quad G'_1 \sqcap G_1[G'_1/X] \leq G'_2 \sqcap G_2[G'_2/X]}{\Omega \vdash \text{pack}\langle G'_1, t_1 \rangle \text{ as } \exists X.G_1 : \exists X.G_1 \leq \text{pack}\langle G'_2, t_2 \rangle \text{ as } \exists X.G_2 : \exists X.G_2} \\ (\leq \text{unpack}) & \frac{\Omega \vdash t_1 : G_1 \leq t_2 : G_2 \quad G_1 \rightarrow \exists X.G'_1 \quad G_2 \rightarrow \exists X.G'_2 \quad \Omega, x : G'_1 \sqsubseteq G'_2 \vdash t'_1 : G'_1 \leq t'_2 : G'_2}{\Omega \vdash \text{unpack}\langle X, x \rangle = t_1 \text{ in } t'_1 : G'_1 \leq \text{unpack}\langle X, x \rangle = t_2 \text{ in } t'_2 : G'_2} \end{aligned}$$

Figure 5.4: $\text{GSF}_\varepsilon^\exists$ and GSF^\exists : Extensions for Strict Precision

$$\begin{aligned} \mathcal{V}_\rho[\exists X.G] = & \{ (W, \varepsilon_1 \text{packu}\langle G_1, v_1 \rangle :: \exists X.\rho(G), \varepsilon_2 \text{packu}\langle G_2, v_2 \rangle :: \exists X.\rho(G)) \in \text{Atom}_\rho^\exists[\exists X.G] \mid \\ & \forall W' \succeq W, \alpha. \exists R \in \text{REL}_{W',j}[G_1, G_2]. (W' \boxtimes (\alpha, G_1, G_2, R), \\ & \varepsilon_1[\hat{G}_1, \hat{\alpha}] v_1 :: \rho(G)[\alpha/X], \varepsilon_2[\hat{G}_2, \hat{\alpha}] v_2 :: \rho(G)[\alpha/X]) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[G] \} \end{aligned}$$

First, we establish that two packages are related if their term components ascribed to the existential type body, substituting the fresh type name α by X , are related. Second, since we ascribed term components to other types, we need evidence justifying this. More specifically, we need two evidences that justify $\rho(G)[G_1/X]$ is consistent with $\rho(G)[\alpha/X]$ and $\rho(G)[G_2/X]$ is consistent with $\rho(G)[\alpha/X]$, respectively. In this sense, we use evidences $\varepsilon_1[\hat{G}_1, \hat{\alpha}]$ and $\varepsilon_2[\hat{G}_2, \hat{\alpha}]$; they are just ε_1 and ε_2 , substituting representation types in the left and the fresh type name α in the right, by X . Note that the combination of these evidences with the internal evidences of the package (term

component evidences) through transitivity can fail.

This interpretation of existential types is pretty complete but is not enough. Now, suppose that we have the packages $\varepsilon_{\exists X. ?} \text{packu} \langle \text{Int}, \varepsilon_{\text{Int}1} :: \text{Int} \rangle :: \exists X. ?$ and $\varepsilon_{\exists X. ?} \text{packu} \langle \text{Bool}, \varepsilon_{\text{Int}1} :: \text{Int} \rangle :: \exists X. ?$. These two packages are very similar; the only difference is in their representation type. They are related under the above interpretation of existential types, due the fact that we can relate $\varepsilon_?(\varepsilon_{\text{Int}1} :: \text{Int})$ and $\varepsilon_?(\varepsilon_{\text{Int}1} :: \text{Int})$ under the unknown type. But we do not want to relate these packages. Note that we could use the packages in the same context (*e.g.* if we ascribe them by the type $\exists X. X$) with different behaviors, losing the property that says if two packages are related, then they are contextually equivalent. Therefore, we need to be more strict in the definition of when two packages are related. Finally, we define the interpretation of existential types as follows:

$$\begin{aligned} \mathcal{V}_\rho \llbracket \exists X. G \rrbracket = & \{ (W, \varepsilon_1 \text{packu} \langle G_1, v_1 \rangle :: \exists X. \rho(G), \varepsilon_2 \text{packu} \langle G_2, v_2 \rangle :: \exists X. \rho(G)) \in \text{Atom}_\rho^= [\exists X. G] \mid \\ & \forall W' \succeq W, \alpha. \exists R \in \text{REL}_{W'.j} [G_1, G_2]. \forall \Xi, \varepsilon \Vdash \Xi; \text{dom}(\rho) \vdash \exists X. G \sim \exists X. G, W' \in \mathcal{S} \llbracket \Xi \rrbracket. \\ & ((\varepsilon_1 \mathbin{\text{\$}} \rho_1(\varepsilon)) \wedge (\varepsilon_2 \mathbin{\text{\$}} \rho_2(\varepsilon))) \Rightarrow (W' \boxtimes (\alpha, G_1, G_2, R), \\ & (\varepsilon_1 \mathbin{\text{\$}} \rho_1(\varepsilon)) \llbracket \hat{G}_1, \hat{\alpha} \rrbracket v_1 :: \rho(G)[\alpha/X], (\varepsilon_2 \mathbin{\text{\$}} \rho_2(\varepsilon)) \llbracket \hat{G}_2, \hat{\alpha} \rrbracket v_2 :: \rho(G)[\alpha/X]) \in \mathcal{F}_{\rho[X \mapsto \alpha]} \llbracket G \rrbracket \} \end{aligned}$$

The representation type of a package in gradual settings act as a pending substitution, which has to make sense for all possible (more precise) existential types. In a static world, we do not have to deal with this problem, because evidence never gains precision, and the initial type checking ensures that the program never fails. For this reason, we extend the interpretation of existential types by quantifying over all evidences that justify that $\exists X. G \sim \exists X. G$. Doing so ensures that the representation type behaves correctly for any existential type that is more precise than $\exists X. G$. Note that studying the encoding of existential into universal types leads us to justify the same definition.

5.5.4 Representation Independence

We prove the soundness of the logical relation extended with existential types with respect to contextual equivalence.

Proposition 5.5 *If $\Xi; \Delta; \Gamma \vdash t_1 \approx t_2 : G$, then $\Xi; \Delta; \Gamma \vdash t_1 \approx^{ctx} t_2 : G$.*

With this result, we can return to the semaphore example and show the representation independence for the two different implementations s_1 and s_3 . Let us recall the definition of these packages, where the former uses `Bool` as representation type, while the latter uses the unknown type:

$$\begin{aligned} s_1 &\equiv \text{pack} \langle \text{Bool}, v_1 \rangle \text{ as Sem} && \text{where } v_1 \equiv \{ \text{bit} = \text{true}, \text{flip} = (\lambda x : \text{Bool}. \neg x), \text{read} = (\lambda x : \text{Bool}. x) \} \\ s_3 &\equiv \text{pack} \langle ?, [v_3] \rangle \text{ as Sem} && \text{where } v_3 \equiv \{ \text{bit} = 1, \text{flip} = (\lambda x. 1 - x), \text{read} = (\lambda x. 0 < x) \} \end{aligned}$$

To prove that these two packages are contextually equivalent (Proposition 5.6), it suffices by Proposition 5.5 to show that each logically approximates the other. (Note that to proceed with the proof below, we deal with the tuple-based representation of `Sem`, since GSF has no records.) We prove only one direction, namely $s_1 \preceq s_3 : \text{Sem}$; the other is proven analogously. Therefore, we are

required to show that $s_1^* \preceq s_3^* : Sem$, where s_1^* and s_3^* are the translation of s_1 and s_3 from GSF^\exists to $\text{GSF}_\varepsilon^\exists$, respectively.

Proposition 5.6 $s_1 \approx^{ctx} s_3 : Sem$

To prove $s_1^* \preceq s_3^* : Sem$, we are required to show that for all W , $(W, s_1^*, s_3^*) \in \mathcal{T}_\emptyset \llbracket Sem \rrbracket$. Therefore, we have to prove that $\vdash s_i^* : Sem$ (but this is already proven) and $(W, s_1^*, s_3^*) \in \mathcal{V}_\emptyset \llbracket Sem \rrbracket$ (since s_i^* are already values). Expanding the definition of $\mathcal{V}_\emptyset \llbracket Sem \rrbracket$, we need to show that $\forall W' \succeq W$ and α , $\exists R \in \text{REL}_{W'.j}[\text{Bool}, ?]$, such that $\forall \varepsilon \Vdash \cdot : \vdash Sem \sim Sem$:

$$(W'', (\varepsilon_{Sem} \circ \varepsilon)[\text{Bool}, \hat{\alpha}]v_1^* :: G[\alpha/X], (\varepsilon_{Sem} \circ \varepsilon)[?, \hat{\alpha}]v_3^* :: G[\alpha/X]) \in \mathcal{T}_{[X \mapsto \alpha]} \llbracket G \rrbracket$$

where $W'' = W' \boxtimes (\alpha, \text{Bool}, ?, R)$, $G = \text{schm}_e^\sharp(Sem) = X \times (X \rightarrow X) \times (X \rightarrow \text{Bool})$, $s_1^* = \varepsilon_{Sem} \text{packu} \langle \text{Bool}, v_1^* \rangle :: Sem$ and $s_2^* = \varepsilon_{Sem} \text{packu} \langle ?, v_2^* \rangle :: Sem$. Since ε_{Sem} is a static evidence, it can not gain precision and so $(\varepsilon_{Sem} \circ \varepsilon) = \varepsilon_{Sem}$. Therefore, now we are required to show

$$(\downarrow W'', v'_1, v'_3) \in \mathcal{V}_{[X \mapsto \alpha]} \llbracket G \rrbracket$$

where

$$\begin{aligned} v'_1 &= \langle \text{Bool} \times (\text{Bool} \rightarrow \text{Bool}) \times (\text{Bool} \rightarrow \text{Bool}), \alpha^{\text{Bool}} \times (\alpha^{\text{Bool}} \rightarrow \alpha^{\text{Bool}}) \times (\alpha^{\text{Bool}} \rightarrow \text{Bool}) \rangle \\ &\quad \langle \text{true}, \langle (\lambda x : \text{Bool}. \neg x), (\lambda x : \text{Bool}. x) \rangle \rangle :: \alpha \times (\alpha \rightarrow \alpha) \times (\alpha \rightarrow \text{Bool}) \\ v'_3 &= \langle \text{Int} \times (? \rightarrow \text{Int}) \times (? \rightarrow \text{Bool}), \alpha^{\text{Int}} \times (\alpha^? \rightarrow \alpha^{\text{Int}}) \times (\alpha^? \rightarrow \text{Bool}) \rangle \\ &\quad \langle 1, \langle (\lambda x. 1 - x), (\lambda x. 0 < x) \rangle \rangle :: \alpha \times (\alpha \rightarrow \alpha) \times (\alpha \rightarrow \text{Bool}) \end{aligned}$$

Taking $R = \{(W^*, \varepsilon_{\text{Bool}} \text{true} :: \text{Bool}, \varepsilon_{\text{Int}} 1 :: ?), (W^*, \varepsilon_{\text{Bool}} \text{false} :: \text{Bool}, \varepsilon_{\text{Int}} 0 :: ?) \mid W^* \succeq W'\}$, it is easy to show that

$$\begin{aligned} &- (\downarrow^2 W'', \langle \text{Bool}, \alpha^{\text{Bool}} \rangle \text{true} :: \alpha, \langle \text{Int}, \alpha^{\text{Int}} \rangle 1 :: \alpha) \in \mathcal{V}_{[X \mapsto \alpha]} \llbracket X \rrbracket \\ &- (\downarrow^2 W'', \langle \text{Bool} \rightarrow \text{Bool}, \alpha^{\text{Bool}} \rightarrow \alpha^{\text{Bool}} \rangle (\lambda x : \text{Bool}. \neg x) :: \alpha \rightarrow \alpha, \\ &\quad \langle ? \rightarrow \text{Int}, \alpha^? \rightarrow \alpha^{\text{Int}} \rangle (\lambda x. 1 - x) :: \alpha \rightarrow \alpha) \in \mathcal{V}_{[X \mapsto \alpha]} \llbracket X \rightarrow X \rrbracket \\ &- (\downarrow^2 W'', \langle \text{Bool} \rightarrow \text{Bool}, \alpha^{\text{Bool}} \rightarrow \text{Bool} \rangle (\lambda x : \text{Bool}. x) :: \alpha \rightarrow \text{Bool}, \\ &\quad \langle ? \rightarrow \text{Bool}, \alpha^? \rightarrow \text{Bool} \rangle (\lambda x. 0 < x) :: \alpha \rightarrow \text{Bool}) \in \mathcal{V}_{[X \mapsto \alpha]} \llbracket X \rightarrow \text{Bool} \rrbracket \end{aligned}$$

Note that $\downarrow^2 W'' \succeq W'$. Thus, the result follows immediately.

5.6 Related Work

New et al. (2020) are the first to formalize and give a logical relation for existential types in a gradual language. In PolyG^ν , programs with both universal and existential types need to be annotated with explicit sealing/unsealing terms in order to typecheck and obtain the desired behavior at runtime. For example, the package s_1 with the existential type $Sem \equiv \exists X. X \times (X \rightarrow X) \times (X \rightarrow \text{Bool})$ from above can be written in PolyG^ν as follows:

$\text{pack}\langle X = \text{Bool}, v_1 \rangle$ where $v_1 \equiv \langle \text{seal}_X(\text{true}), \langle \lambda x : X. \text{seal}_X(\neg \text{unseal}_X(x)), \lambda x : X. \text{unseal}_X(x) \rangle \rangle$

Note that a package in PolyG^ν does not need to be annotated with an existential type since the package operations already have a polymorphic type. For example, the function’s implementation to flip the state of the semaphore has type $X \rightarrow X$, while in our s_1 implementation, it has type $\text{Bool} \rightarrow \text{Bool}$. It is worth mentioning that the existential types of PolyG^ν will have the same downsides discussed in Section 2.4 for its universal types.

5.7 Conclusion

This chapter extends GSF with support for existential types for gradual data abstraction. The extension of GSF with existential types is also derived using AGT, and similar adjustments to those implemented by GSF have to be done to recover parametricity, thus satisfying parametricity and a weak dynamic gradual guarantee.

The next part proposes plausible sealing as a new intermediate language mechanism that satisfies both parametricity and graduality.

Part II

Plausible Sealing for Gradual Parametricity

Introduction

As we saw in Chapter 3, deriving a gradual version of System F that complies with parametricity and graduality has been and continues to be a challenging problem. On the one hand, we have the languages that base their design on type-driven sealing like λB , System F_G , and GSF that preserve the familiar syntax of System F but violate graduality to a greater or lesser extent. On the other hand, we have PolyG' , a term-driving sealing language that fulfills parametricity and graduality but at the cost of forgetting the familiar syntax of System F and with its own limitations (Section 2.4).

The difficulty observed in the type-driven proposals is a strong tension between the two desirable properties. Unfortunately, when applying a polymorphic function with an imprecise type, the decision of whether arguments should be sealed or not is not so clear-cut. Consider, for example, the functions $f_1 = \Lambda X. \lambda x : ?. x :: X$ and $f_2 = \Lambda X. \lambda x : ?. x :: \text{Int}$. By graduality, the two functions should behave like their more precisely-typed versions $\Lambda X. \lambda x : X. x :: X$ and $\Lambda X. \lambda x : \text{Int}. x :: \text{Int}$, respectively. However, this means that applying both functions to type Int and value 42 should treat their arguments differently even though they have the same parameter type. Applying $f_1 [\text{Int}] 42$ should seal the argument 42, while $f_2 [\text{Int}] 42$ should not. The type-driven sealing approach decides whether to seal or not based on the types of the applied function and the argument. However, there is no way to make this choice *a priori* and modularly, without breaking graduality. For example, GSF does not seal the argument here, breaking graduality for f_1 .

In this part, we revisit the original problem: gradual parametricity with the familiar syntax of System F. Consider again the applications $f_1 [\text{Int}] 42$ and $f_2 [\text{Int}] 42$. Instead of making an arbitrary choice between sealing or not sealing, we propose to keep both options open, so the decision can be made when the value 42 is actually used. This novel technique, called *plausible sealing*, essentially allows our calculus to treat the applications as $f_i [\text{Int}] (\text{maybeSeal}_X(42))$. The maybe-sealed value 42 embeds the fact that it may be both sealed at X and unsealed, which makes the two applications successfully reduce to 42. To study plausible sealing, we propose in Chapter 6 an intermediate gradual parametric language, Funky ($F_\varepsilon^?$), which can be used as the elaboration target of different gradual source languages. We describe in Chapter 7 one such source language, Funk ($F^?$), with the familiar syntax of System F.² It is worth pointing out that for simplicity, we formalize the approach in a setting where polymorphism is limited to instantiations with base and variable types.

The key novelty of the intermediate language $F_\varepsilon^?$ is that it introduces maybe-sealing forms, which are interpreted thanks to an innovative runtime tracking technique. Additionally, $F_\varepsilon^?$ avoids the use of dynamically-generated *global* seals. In previous calculi, a seal α can continue to exist when the type variable X for which it was created goes out of scope: $(\Lambda X. \lambda x : X. x :: ?) [\text{Int}] 42 \rightarrow^* \text{seal}_\alpha(42)$. In fact, seals in these calculi behave as a form of symbolic cryptography, which makes it possible to embed languages with runtime sealing (Pierce and Sumii, 2000; Sumii and Pierce, 2004) (Chapter 4). But at the same time, global seals have been shown to break equivalences that hold in System F (Devriese et al., 2018). This global nature of seals is also the reason that parametricity theorems for gradual calculi so far have used formulations based on Kripke worlds containing semantic types for dynamically-allocated seals (Jacobs et al., 2021). $F_\varepsilon^?$ features *lexically-scoped sealing*, and it is the first to support a stronger formulation of parametricity where semantic types are tracked in a lexical environment, similar to traditional formulations of parametricity (Reynolds, 1983). As such, $F_\varepsilon^?$ could perhaps satisfy the ambitious criterion for gradual languages recently proposed by Jacobs

² Funk is for **F-unknown** ($F^?$), and Funky is for Funk with **evidence** ($F_\varepsilon^?$).

et al. (2021): fully abstract embedding of the statically-typed language into the gradually-typed language. This has been disproved by Devriese et al. (2018) for λB , but their counterexample, which essentially relies on the global nature of seals in λB and GSF, does not apply to $F_\varepsilon^?$. Finally, we prove both graduality and parametricity for the intermediate language $F_\varepsilon^?$.

The elaboration of a source language like $F^?$ to $F_\varepsilon^?$ must be in charge of introducing maybe-sealing forms when imprecise types occur in type applications. For $F^?$, we establish graduality, currently subject to a restriction on type applications which will be explained in detail in Chapter 7. In addition to graduality, we explain the source-level parametric reasoning that $F^?$ offers.

It is worth noting that parametric reasoning at the source level of a gradual language is subtle because of another point of tension between parametricity and gradual typing that was pointed out by New et al. (2020). Consider the two applications: $(\Lambda X. \lambda x : ?. x :: X) [\text{Int}] 42$ and $(\Lambda X. \lambda x : ?. x :: X) [\text{Bool}] 42$. Since the behavior of the polymorphic function $\Lambda X. \lambda x : ?. x :: X$ should not depend on the type it is applied to, a strict interpretation of parametricity dictates that both applications should behave the same. At the same time, by graduality, the first application should behave equivalently to the following more precisely typed version, which reduces to 42:

$$(\Lambda X. \lambda x : X. x :: X) [\text{Int}] 42 \rightarrow^* 42$$

However, the second application is of type `Bool` and there is no reasonable way to come up with a boolean value to return. Even worse, because parametricity implies preservation of relatedness of values, successfully returning a boolean in the second application would imply a contradiction, because that boolean would have to be related to 42 in an arbitrary, caller-chosen relation, even when that relation is empty. In other words, this strict interpretation of source-level parametricity is incompatible with graduality. However, that is not the end of the story.


In $F^?$, the second application fails at runtime: the value 42 does not have the right type to be sealed at type X , so it is not maybe-sealed, and we simply report an error when it is treated as a value of type X . This means that some polymorphic $F^?$ terms may behave differently depending on the type they are applied to, as we have $(\Lambda X. \lambda x : ?. x :: X) [\text{Int}] 42 \mapsto^* 42$ and $(\Lambda X. \lambda x : ?. x :: X) [\text{Bool}] 42 \mapsto^* \text{error}$. It would however be incorrect to conclude that $F^?$ is not parametrically polymorphic. First, uniformity of behavior is satisfied for polymorphic functions of fully precise types,³ even if they internally use type applications that do (!). In these cases, the definition of parametricity coincides with the standard definition for System F—except that related terms may also simultaneously fail with a runtime type error. In other words, the differences in behavior can only occur for imprecise types (and can therefore be avoided using ascriptions to precise types). Intuitively, these differences are a consequence of $F^?$ applying plausible sealing in an attempt to infer whether the programmer intended to treat arguments (or results) as values of the quantified type X , in a maximally permissive way. However, the behavior of plausible sealing is entirely predictable based on type information available *statically* at the call site, and does not depend on runtime type information. When one takes this behavior into account, gradual parametricity in $F^?$ still implies useful free theorems. For example, for any $f : \forall X. ? \rightarrow X$, $f [\text{Bool}] \text{true}$ may diverge, fail or return the value `true`, but it can never return `false`.

³ Later on, we introduce a mechanism to annotate occurrences of the unknown type with the subset of type variables in scope that it might denote, and explain the impact of this feature on parametric reasoning for imprecise types.

Chapter 6

The Gradual Language for Plausible Sealing $F_\varepsilon^?$

In this chapter, we present the target language $F_\varepsilon^?$. We develop a novel approach to gradual parametricity based on plausible sealing. Technically, we use lexically-scoped rather than global sealing, and a novel runtime tracking mechanism based on proof-relevant precision to account for postponing sealing decisions. This is achieved using a representation of evidence inspired by the category of spans. Focusing on the new ideas, we formally develop our approach in a simplified setting where polymorphism is restricted to instantiations with base and variable types. We prove that the proposed intermediate language $F_\varepsilon^?$ satisfies both parametricity and graduality, and mechanize the two key lemmas in Agda needed to prove these properties. We illustrate the practicality of $F_\varepsilon^?$ by providing a translation from the source gradual language $F^?$ (Chapter 7).

Overview. In Section 6.1, we illustrate the behavior of $F_\varepsilon^?$ programs by starting from their $F^?$ source counterparts, and compare to other approaches. We then formalize the core calculus $F_\varepsilon^?$ (Section 6.2), describe its novel form of runtime tracking mechanism for plausible sealing (Section 6.3), and prove parametricity (Section 6.4) and the gradual guarantees (Section 6.5). Section 6.6 discusses related work and Section 6.7 concludes. Full definitions and proofs of the main results can be found in the Appendix. Also, we develop mechanized proofs of two key technical results in Agda (Lemmas 6.7 and 6.12, marked with ) . The implementation (<https://doi.org/10.5281/zenodo.6341550>) exhibits typing derivations, the translation from $F^?$ to $F_\varepsilon^?$, and reduction traces, including all the examples mentioned in this work and of the related literature.

6.1 Overview of $F_\varepsilon^?$

This section outlines the behavior of $F_\varepsilon^?$ with specific source program examples in $F^?$ from the current state of the art of gradual parametricity, informally shedding light on how plausible sealing is realized and compares to other approaches. We use the **blue color** and sans serif fonts for source languages and the **red color** and bold fonts for target languages.

6.1.1 Evidence for Plausible Sealing

The concept of evidence is very general and applies to a variety of typing disciplines. Garcia et al. (2016) observe that for a language with only type consistency, evidence coincides with the middle type of threesomes (Siek and Wadler, 2010). A threesome is a three-place cast, $\langle \mathbf{G}_2 \xleftarrow{\mathbf{G}} \mathbf{G}_1 \rangle$, representing a downcast from the source type \mathbf{G}_1 to the middle type \mathbf{G} , followed by an upcast from the middle type to the target type \mathbf{G}_2 . This representation allows for space efficiency of cast calculi: when combining two threesomes, it is sufficient to retain the outermost types and keep the *meet* \sqcap (according to the precision partial order) of the middle types. If such a meet is not defined, the combination of threesomes fails with a cast error. For instance, the combination $\langle \mathbf{Int} \xleftarrow{\mathbf{Int}} ? \rangle \langle ? \xleftarrow{\mathbf{Bool}} \mathbf{Bool} \rangle$ fails because $\mathbf{Int} \sqcap \mathbf{Bool}$ is undefined.

In this work, we adopt AGT for deriving the static semantics of $F^?$ and $F_\varepsilon^?$, and define the dynamic semantics of $F^?$ by elaboration to the evidence-based target language $F_\varepsilon^?$. As Chapter 3 shows, using AGT blindly for gradual parametricity only ensures type safety for the obtained semantics, but not parametricity. Ensuring parametricity requires a refined representation of evidence and consistent transitivity. In GSF, evidence is represented not as a single type, but as a pair of types (extended with type names tracked globally), in order to capture the directionality of consistent judgments, which can intuitively denote either *sealing* or *unsealing*. Consistent transitivity is refined to forbid unsound unsealing and hence enforce parametricity. In order to address the limitations discussed in the introduction, we design a novel representation of evidence in $F_\varepsilon^?$, to realize plausible sealing. The rest of this section informally describes this novel representation of evidence and the achieved behavior.

Let us focus on the two terms (1) f_1 [Int] 42 and (2) f_2 [Int] 42 used in the introduction. As explained, these are key illustrations of the challenge of type-driven sealing: any early decision to either seal or not seal the argument would make one of these examples fail, thereby breaking graduality. Our approach consists of capturing the different possibilities regarding sealing, and postponing the choice to seal or not to seal until a value is used; as a consequence, both programs successfully reduce to 42. This is achieved by the evidence, which accommodates the different valid usages of an argument of unknown type whenever the unknown type has in its scope some type variable. For this reason, the first step consists of decorating the unknown type with the type variables that are in scope. So the type of both elaborated polymorphic functions in $F_\varepsilon^?$ are $\forall \mathbf{X}. ?_{\mathbf{X}} \rightarrow \mathbf{X}$ and $\forall \mathbf{X}. ?_{\mathbf{X}} \rightarrow \mathbf{Int}$, respectively, since X is the only type variable in the scope of the unknown type. The argument 42 is of type \mathbf{Int} , so upon elaboration an ascription to $?$ is introduced—there are no type variables in scope at that point. Hence, the elaboration of both examples (where \mathbf{G} stands for either \mathbf{X} (1) or \mathbf{Int} (2)) is:

$$(\varepsilon_1 ((\lambda \mathbf{X}. \lambda \mathbf{x} : ?_{\mathbf{X}}. \varepsilon_1 \mathbf{x} :: \mathbf{G}) [\mathbf{Int}]) :: ? \rightarrow \mathbf{Int}) (\varepsilon_2 42 :: ?)) \quad (6.1)$$

When these polymorphic functions are instantiated at type \mathbf{Int} , the decorations of unknown types are enriched with the instantiation information, so the lambda-abstractions both take an argument of type $?_{\mathbf{X}:\mathbf{Int}}$. To proceed with the beta reduction, the argument $\varepsilon_2 42 :: ?$ is ascribed to the expected argument type of the lambda, yielding the value $\mathbf{v} = \varepsilon' 42 :: ?_{\mathbf{X}:\mathbf{Int}}$. This value is the *maybeSeal_X*(42) used in the introduction. Observe that there are *two* ways in which the type of 42, \mathbf{Int} , is consistent with $?_{\mathbf{X}:\mathbf{Int}}$: either because $?_{\mathbf{X}:\mathbf{Int}}$ stands for \mathbf{Int} , or because it stands for \mathbf{X} (which happens to be instantiated with \mathbf{Int}). So it is *plausible* that the value be sealed at type X , though not mandatory. In order to account for this multiplicity of possibilities, we let ε'

Table 6.1: Comparisons of gradual parametricity approaches.

	Source term in $F^?$	$F_\varepsilon^?$	λB	System F_G	GSF	PolyG $^\nu$
1	$(\lambda X.\lambda x:?.x::X) \text{ [Int] } 42$	42	error	error	error	error / 42
2	$(\lambda X.\lambda x:?.x::\text{Int}) \text{ [Int] } 42$	42	42	42	42	42 / error
3	$(\lambda X.\lambda x:?.x::X) \text{ [Bool] } 42$	error	error	error	error	error
4	$((\lambda X.\lambda x:X.x::?) \text{ [Int] } 42) + 1$	43	error	error	error	error / 43
5	$(\lambda X.\lambda x:X.(x::?) + 1) \text{ [Int] } 3$	error	error	error	error	error
6	$(\lambda X.\lambda Y.\lambda x:?.(x,x)::X \times Y) \text{ [Int] [Int] } 42$	$\langle 42, 42 \rangle$	error	error	error	error / error

be a *set* of justifications, rather than a single justification as is standard in AGT (and in GSF). Both justifications support the same consistency judgment $\text{Int} \sim ?_{X:\text{Int}}$, so using just the meet is insufficient. Instead, we represent a justification of a consistent judgment between types G_1 and G_2 as a triple (G, c_1, c_2) , where G is the meet, and c_1 (resp. c_2) is a proof term that characterizes *how* G is more precise than G_1 (resp. G_2). Hence, precision in $F_\varepsilon^?$ is a *proof-relevant* notion, and evidences carry these proofs. In the example, the precision judgments are $\text{inj}_X : \text{Int} \sqsubseteq ?_{X:\text{Int}}$ and $\text{inj}_{\text{Int}} : \text{Int} \sqsubseteq ?_{X:\text{Int}}$, where the proof terms inj_X and inj_{Int} denote the two possible injections of imprecision. We write refl_{Int} for the proof term of $\text{Int} \sqsubseteq \text{Int}$. So we have:

$$\varepsilon' = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_X), (\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\text{Int}})\}$$

When \mathbf{v} is substituted in the body, reduction proceeds by combining ε' with ε , the evidence inserted by the elaboration of the ascription in the body (Equation 6.1), using consistent transitivity. Importantly, in Example (1), ε justifies that the unknown type is consistent with X via inj_X , and when Int is substituted for X , the proof term inj_X in ε does not change (although it now justifies the judgment $\text{Int} \sqsubseteq ?_{X:\text{Int}}$ rather than $X \sqsubseteq ?_X$). Then reduction proceeds by checking that there is at least one justification in ε' that is compatible with ε ; otherwise an error is raised. Because such a justification exists in both examples, they both successfully reduce to 42.

In essence, we treat type type precision \sqsubseteq in $F_\varepsilon^?$ not simply as a preorder, but as a category, and we construct evidence as a variant of the category of spans. Spans are the triples (G, c_1, c_2) , and evidences are sets of spans. Composition of evidence through consistent transitivity can then be defined in terms of a category-theoretic pullback operation, again generalizing the order-theoretic meet that is used in regular threesomes and AGT.

6.1.2 Comparing Plausible Sealing and Prior Approaches

We now outline the behavior of $F_\varepsilon^?$, informally shedding light on how plausible sealing is realized and compares to other approaches. For the sake of simplicity and understanding, we use source $F^?$ programs for the comparison. Note that, in order to be well typed, source terms in $F^?$ need to be augmented with evidence in $F_\varepsilon^?$, casts in λB , and seal/unseal terms in PolyG $^\nu$ (possibly yielding two possible well-typed variants), in addition to superficial syntactic differences. Table 6.1 compares $F_\varepsilon^?$ with prior approaches using a number of key examples from the literature—except Example (6)—either adapted or verbatim. Additional examples are provided in the Appendix.

Examples (1) and (2) are the key examples discussed in Section 6.1.1. In GSF, λB and System F_G , Example (1) fails with an error, and Example (2) yields 42, because these systems eagerly

choose not to seal the argument when it has the unknown type. In PolyG^ν , programmers have to use explicit sealing to decide to seal or not, but this cannot be done modularly; one can obtain different behaviors accordingly. Example (3) raises a runtime error at the ascription to \mathbf{X} , as the type variable is instantiated to `Bool` but a value of type `Int` is provided. Note that other approaches also raise an error in this example because the argument is not sealed, which implies that Example (1) fails as well. Example (4) illustrates that, contrary to other approaches that use global type names as a runtime sealing mechanism, sealing in $F_\varepsilon^?$ is lexically scoped: seals cannot outlive the lexical boundary of a type abstraction. In the example, when 42 is returned by the function, it is automatically unsealed and usable as a regular integer. In PolyG^ν , an explicit unseal is needed to avoid failure. Example (5) illustrates the prevention of a violation of parametricity at runtime. Example (6) illustrates yet another flexibility of plausible sealing that makes it more expressive than prior approaches: evidence as sets of spans can support multiple sealing behaviors. In this example, the argument of the function, 42, is treated as plausibly sealed to both \mathbf{X} and \mathbf{Y} at the same time. This example fails in GSF. In PolyG^ν , programmers have to pick in advance whether to seal with \mathbf{X} or \mathbf{Y} , and the example fails in both cases. Observe that this program does not have a fully statically-typed counterpart, and therefore showcases an expressiveness gain of the gradual language, which compromises neither graduality nor parametricity.

These examples illustrate the flexibility afforded by plausible sealing, as a novel point in the design space of gradual parametricity.

6.2 The Evidence-Based Language $F_\varepsilon^?$

Now that we have informally explained our representation of evidence and the obtained behavior, we turn to the formalization of $F_\varepsilon^?$ and its properties: parametricity and graduality. This section centers on presenting the language without entering into the details of evidence: evidence and its operators are treated abstractly. We provide the full details of evidence for $F_\varepsilon^?$ in Section 6.3. Sections 6.4 and 6.5 establish parametricity and graduality of $F_\varepsilon^?$, respectively.

6.2.1 Syntax and static semantics

Figure 6.1 presents the syntax and semantics of $F_\varepsilon^?$. A type \mathbf{G} can be either a base type, a type variable, a function type, a pair type, a polymorphic type, or the unknown type. Observe that static types from System F are syntactically included in gradual types \mathbf{G} . In $F_\varepsilon^?$, polymorphic types can only be instantiated with base types and type variables, called *instantiation types*, and denoted by metavariable \mathbf{F} . As mentioned in the introduction, this restriction on polymorphism simplifies the already-dense technical development while still manifesting all the subtleties of gradual parametricity identified in prior work. Another distinctive feature of $F_\varepsilon^?$ is that it avoids the use of a global typename store as used in all prior work on gradual parametricity thanks to the fact that the unknown type is indexed by an environment δ . This instantiation environment keeps track of the static and dynamic information related to type variables in scope: $?_{\mathbf{X}:\text{Int}}$ expresses that type variable \mathbf{X} is in scope and instantiated to `Int`. Uninstantiated type variables are associated with themselves $\mathbf{X} : \mathbf{X}$, which for brevity we simply write as \mathbf{X} . It is worth noting that in the type $?_{\mathbf{X}:\mathbf{X}}$, the two occurrences of \mathbf{X} play a different role: the first is merely a label, while the second is an actual occurrence of the type variable \mathbf{X} .

A term \mathbf{t} can be a value \mathbf{v} , a variable, a term application, a pair, a projection, a type application

$X \in \text{TYPEVAR}$, $G \in \text{GTYPE}$, $\varepsilon \in \text{EVIDENCE}$, $t \in \text{TERM}$, $\Delta \subset \text{TYPEVAR}$, $\Gamma \in \text{VAR} \stackrel{\text{fin}}{\text{GTYPE}}$

$F ::= B \mid X \quad G ::= B \mid X \mid G \rightarrow G \mid G \times G \mid \forall X.G \mid ?_\delta \quad \delta ::= \delta, X : F \mid \emptyset$
 $u ::= b \mid \lambda x : G.t \mid \langle u, u \rangle \mid \Lambda X.t$
 $v ::= \varepsilon u :: G \quad t ::= v \mid x \mid t t \mid \langle t, t \rangle \mid \pi_i(t) \mid t [F] \mid \varepsilon t :: G \quad s ::= u \mid t$

$\Delta; \Gamma \vdash s : G$ Term typing

$$\begin{array}{c} \text{Gb} \frac{\theta(b) = B \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash b : B} \qquad \text{G}_\lambda \frac{\Delta; \Gamma, x : G_1 \vdash t : G_2}{\Delta; \Gamma \vdash \lambda x : G_1.t : G_1 \rightarrow G_2} \\ \text{G}_\times \frac{\Delta; \Gamma \vdash s_1 : G_1 \quad \Delta; \Gamma \vdash s_2 : G_2}{\Delta; \Gamma \vdash \langle s_1, s_2 \rangle : G_1 \times G_2} \qquad \text{G}_\Lambda \frac{\Delta, X; \Gamma \vdash t : G \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \Lambda X.t : \forall X.G} \\ \text{G}_x \frac{x : G \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x : G} \qquad \text{G}_{\text{asc}} \frac{\Delta; \Gamma \vdash s : G' \quad \Delta \vdash G \quad \varepsilon : G' \sim G}{\Delta; \Gamma \vdash \varepsilon s :: G : G} \\ \text{G}_{\text{app}} \frac{\Delta; \Gamma \vdash t_1 : G_1 \rightarrow G_2 \quad \Delta; \Gamma \vdash t_2 : G_1}{\Delta; \Gamma \vdash t_1 t_2 : G_2} \qquad \text{G}_{\text{pairi}} \frac{\Delta; \Gamma \vdash t : G_1 \times G_2}{\Delta; \Gamma \vdash \pi_i(t) : G_i} \\ \text{G}_{\text{appG}} \frac{\Delta; \Gamma \vdash t : \forall X.G \quad \Delta \vdash F}{\Delta; \Gamma \vdash t [F] : G[F/X]} \end{array}$$

$t \rightarrow t$ or error Notion of reduction

$$\begin{array}{l} \text{(Rasc)} \quad \varepsilon_2(\varepsilon_1 u :: G_1) :: G_2 \rightarrow \begin{cases} \varepsilon u :: G_2 & \text{if } \varepsilon = \varepsilon_1 \circ \varepsilon_2 \\ \text{error} & \text{otherwise} \end{cases} \\ \text{(Rapp)} \quad \begin{array}{l} (\varepsilon_1(\lambda x : G_{11}.t) :: G_1 \rightarrow G_2) \\ (\varepsilon_2 u :: G_1) \end{array} \rightarrow \begin{cases} \text{cod}(\varepsilon_1)(t[(\varepsilon u :: G_{11})/x]) :: G_2 & \text{if } \varepsilon = \varepsilon_2 \circ \text{dom}(\varepsilon_1) \\ \text{error} & \text{otherwise} \end{cases} \\ \text{(Rpair)} \quad \langle \varepsilon_1 u_1 :: G_1, \varepsilon_2 u_2 :: G_2 \rangle \rightarrow (\varepsilon_1 \times \varepsilon_2) \langle u_1, u_2 \rangle :: G_1 \times G_2 \\ \text{(Rproji)} \quad \pi_i(\varepsilon \langle u_1, u_2 \rangle :: G_1 \times G_2) \rightarrow p_i(\varepsilon) u_i :: G_i \\ \text{(RappG)} \quad (\varepsilon(\Lambda X.t) :: \forall X.G) [F] \rightarrow (\text{schm}(\varepsilon) t :: G) [F/X] \end{array}$$

$t \mapsto t$ or error Evaluation frames and reduction

$$f ::= \varepsilon \square :: G \mid \square t \mid v \square \mid \langle \square, t \rangle \mid \langle v, \square \rangle \mid \pi_i(\square) \mid \square [F]$$

$$\text{R}_{\rightarrow} \frac{t \rightarrow t'}{t \mapsto t'} \qquad \text{R}_f \frac{t \mapsto t'}{f[t] \mapsto f[t']} \qquad \text{R}_{\text{err}} \frac{t \rightarrow \text{error}}{t \mapsto \text{error}} \qquad \text{R}_{\text{ferr}} \frac{t \mapsto \text{error}}{f[t] \mapsto \text{error}}$$

Figure 6.1: $F_\varepsilon^?$: Syntax, Static and Dynamic Semantics.

(to an instantiation type), or an ascription. Note the presence of an evidence ε in an ascription, to justify the fact that the underlying term is of a type consistent with the ascribed type. Values \mathbf{v} are ascribed raw values $\varepsilon \mathbf{u} :: \mathbf{G}$, where ε justifies that the type of \mathbf{u} is consistent with \mathbf{G} . A raw value \mathbf{u} can be a base value \mathbf{b} , a function, a pair of raw values, or a type abstraction. To avoid duplication of typing rules, we use metavariable \mathbf{s} to denote both raw values \mathbf{u} and terms \mathbf{t} .

The typing judgment $\Delta; \Gamma \vdash \mathbf{s} : \mathbf{G}$ establishes that \mathbf{s} has type \mathbf{G} , under type variable environment Δ , and type environment Γ . Δ is used to track type variables in scope, and Γ to map variables to their types. Most of the type rules are standard, closely following System F. Note that rule (**Gasc**) is the only rule that uses the consistency relation; all other elimination rules require types to match exactly. Elaboration from the source language $F^?$ is in charge of introducing the necessary ascriptions to safely support the flexibility of gradual typing. Rule (**GappG**) is almost standard, save for the fact that it restricts instantiations to instantiation types \mathbf{F} . The type substitution operator $\mathbf{G}[\mathbf{F}/\mathbf{X}]$ is standard, except for occurrences of unknown types, for which type substitution is applied to their instantiation environments δ : $(\delta, \mathbf{X} : \mathbf{F}')[\mathbf{F}/\mathbf{X}] = \delta[\mathbf{F}/\mathbf{X}], \mathbf{X} : \mathbf{F}'[\mathbf{F}/\mathbf{X}]$. For instance, $?_{\mathbf{Y}:\mathbf{X},\mathbf{X}:\mathbf{X}}[\mathbf{Int}/\mathbf{X}] = ?_{\mathbf{Y}:\mathbf{Int},\mathbf{X}:\mathbf{Int}}$. Notice that type substitution on an instantiation environment δ only affects type variable occurrences, not labels.

6.2.2 Dynamic semantics

The dynamic semantics of $F^?$ are usual for an evidence-based reduction semantics (Garcia et al., 2016), using reduction frames and notions of reduction.¹ Reduction uses the consistent transitivity operator \ddagger to combine evidence and justify transitive judgments. If $\varepsilon_1 \ddagger \varepsilon_2$ is defined then it yields a more precise evidence, otherwise an error is raised. For example, rule (**Rasc**) reduces nested ascriptions, such as value $\varepsilon_1 \mathbf{u} :: \mathbf{G}_1$ ascribed to \mathbf{G}_2 using evidence ε_2 . Recall that ε_1 justifies that \mathbf{G}_u , the type of \mathbf{u} , is consistent with \mathbf{G}_1 , noted $\varepsilon_1 : \mathbf{G}_u \sim \mathbf{G}_1$, and likewise, $\varepsilon_2 : \mathbf{G}_1 \sim \mathbf{G}_2$. Therefore, if $\varepsilon_1 \ddagger \varepsilon_2$ is defined, then the resulting evidence justifies the transitive judgment between \mathbf{G}_u and \mathbf{G}_2 , i.e. $\varepsilon_1 \ddagger \varepsilon_2 : \mathbf{G}_u \sim \mathbf{G}_2$. Rule (**Rapp**) reduces a term application substituting the argument in the body of the function. It first ascribes the argument to \mathbf{G}_{11} , the type of \mathbf{x} . To justify the transitive judgment of the beta reduction, it combines ε_2 , with $dom(\varepsilon_1)$. Evidence $dom(\varepsilon_1)$ and $cod(\varepsilon_1)$ can be extracted from ε_1 by reasoning about inversion on consistency ($\varepsilon_1 : \mathbf{G}_{11} \rightarrow \mathbf{G}_{12} \sim \mathbf{G}_1 \rightarrow \mathbf{G}_2$). Evidence ε_1 justifies that $\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}$, the underlying type of the function, is consistent with $\mathbf{G}_1 \rightarrow \mathbf{G}_2$. Thus, evidence $dom(\varepsilon_1)$ justifies that \mathbf{G}_1 is consistent with \mathbf{G}_{11} , and therefore $\varepsilon \ddagger dom(\varepsilon_1)$, if defined, justifies that the type of the raw value \mathbf{u} is consistent with \mathbf{G}_{11} . The output of the function is ascribed to the expected return type \mathbf{G}_2 using the co-domain evidence $cod(\varepsilon_1)$. Rules related to pairs are defined analogously. Rule (**RappG**) reduces type application by substituting type \mathbf{F} in the schema evidence $schm(\varepsilon)$, in the body of the type abstraction \mathbf{t} and in the scheme type \mathbf{G} . By inversion on consistency, if $\varepsilon : \forall \mathbf{X}. \mathbf{G}' \sim \forall \mathbf{X}. \mathbf{G}$, then $schm(\varepsilon) : \mathbf{G}' \sim \mathbf{G}$. Substitution on evidence $\varepsilon[\mathbf{F}/\mathbf{X}]$ is defined using substitution on types, for all type information that appears in the evidence (Section 6.3). Substitution on terms $\mathbf{t}[\mathbf{F}/\mathbf{X}]$ is recursively defined over subterms, evidences, and types.

It is worth noting that rule (**RappG**) is remarkably standard unlike other gradual polymorphic calculi where dynamic type generation happens in this rule, being stored in a global store. This is made possible thanks to the use of the annotated unknown type $?_\delta$. Another point that has relevance in the reduction of type applications is that the type of the redex can contain instantiated

¹The dynamic semantics of $F^?$ are very similar to GSF_ε , except for the transitivity operator definition and the reduction rule for type applications.

type variables in scope. For example, term $(\varepsilon_2 (\Lambda X. \lambda x : ?_X. \varepsilon_1 x :: X) :: \forall X. ?_X \rightarrow X)$ $[\mathbf{Int}]$ has type $?_{X:\mathbf{Int}} \rightarrow \mathbf{Int}$ with X in the instantiation environment of the unknown type. To obtain a term that can be applied to an argument of type $?_\emptyset$, an external evidence to the application is necessary to justify that $?_{X:\mathbf{Int}} \rightarrow \mathbf{Int}$ is consistent with $?_\emptyset \rightarrow \mathbf{Int}$. As we saw in Section 6.1 and will explain in detail in Chapter 7, this evidence is inserted by the elaboration from $F^?$ to $F_\varepsilon^?$.

Properties. As expected from any language, $F_\varepsilon^?$ is type safe (*i.e.* well-typed $F_\varepsilon^?$ terms do not get stuck). Thus, a well-typed program either evaluates to a value, a runtime error, or diverges. In order to prove type safety, it is necessary to have some properties about the evidence, such as the resulting evidence from consistent transitivity supports the transitive consistency judgment and type substitution over the evidence supports the substitution over the judgment (Section 6.3).

Lemma 6.1 (Type Safety) *If $\vdash t : \mathbf{G}$ then either $t \mapsto^* v$ with $\vdash v : \mathbf{G}$, $t \mapsto^* \mathbf{error}$, or t diverges.*

Of course, the most interesting properties of $F_\varepsilon^?$ are parametricity and graduality. We dive into the details of these properties in Section 6.4 and Section 6.5 respectively, after giving a detailed account of evidence, including its representation, operations, and properties thereof, in particular *associativity* and *monotonicity* of consistent transitivity.

6.3 Evidence for Plausible Sealing in $F_\varepsilon^?$

We now turn to the key technical innovation that makes $F_\varepsilon^?$ (and by extension, $F^?$) able to address the dilemma presented at the beginning of this Chapter: plausible sealing, implemented via a novel representation of evidence based on a *proof-relevant* notion of gradual type precision. As explained in Section 6.1, for a consistency judgment $\mathbf{G}_1 \sim \mathbf{G}_2$, instead of having evidence only track a common more precise type \mathbf{G} , evidence is a set of spans, where each span includes a common more precise type \mathbf{G} and two proof terms that describe *how* $\mathbf{G} \sqsubseteq \mathbf{G}_1$ and $\mathbf{G} \sqsubseteq \mathbf{G}_2$ hold, respectively.

6.3.1 Proof-relevant precision

As mentioned in Section 6.1.1, there can be multiple ways of satisfying a precision relation $\mathbf{G} \sqsubseteq \mathbf{G}'$. To differentiate them, we extend the precision relation between types with a proof term \mathbf{c} that expresses how \mathbf{G} is more precise than \mathbf{G}' .

Proof-relevant precision is presented in Figure 6.2. The proof relevant judgment $\mathbf{c} : \mathbf{G} \sqsubseteq \mathbf{G}'$ denotes that *proof term* \mathbf{c} justifies that \mathbf{G} is more precise than \mathbf{G}' . A reflexive proof term \mathbf{refl}_B justifies that B is more precise than B . Similarly, \mathbf{refl}_X justifies that X is more precise than X . A function proof term $\mathbf{c} \rightarrow \mathbf{c}'$ witnesses that a function type is more precise than another function type if their domains and codomains are related; likewise for pair $(\mathbf{c} \times \mathbf{c})$ and polymorphic proof terms $(\forall X. \mathbf{c})$. Proof term \mathbf{inj}_X represents an injection from X into $?_\delta$, and witnesses that if X is associated to F in δ and F is well-formed with respect to δ , then F is more precise than $?_\delta$. We say that a type F is well-formed with respect to δ ($\delta \vdash F$) if F is a base type B or is a type variable X and $X : X \in \delta$. For example, if X is not yet instantiated and $X : X \in \delta$, then $\mathbf{inj}_X : X \sqsubseteq ?_{X:X}$. If $X : \mathbf{Int} \in \delta$, then $\mathbf{inj}_X : \mathbf{Int} \sqsubseteq ?_{X:\mathbf{Int}}$. Injection \mathbf{inj}_B witnesses that a base type B is more precise than any unknown type. Proof term sequence $\mathbf{inj}_{\rightarrow}(\mathbf{c})$ justifies that function types are more

$\mathbf{c} ::= \text{refl}_B \mid \text{refl}_X \mid \mathbf{c} \rightarrow \mathbf{c} \mid \mathbf{c} \times \mathbf{c} \mid \forall X. \mathbf{c} \mid \text{inj}_B \mid \text{inj}_X \mid \text{inj}_{\rightarrow}(\mathbf{c}) \mid \text{inj}_{\times}(\mathbf{c}) \mid \text{inj}_{\forall}(\mathbf{c}) \mid \text{inj}_{?}$

$\boxed{\mathbf{c} : \mathbf{G} \sqsubseteq \mathbf{G}}$ Proof-relevant precision

$$\begin{array}{c}
\frac{}{\text{refl}_B : B \sqsubseteq B} \quad \frac{}{\text{refl}_X : X \sqsubseteq X} \quad \frac{\delta \sqsubseteq \delta'}{\text{inj}_{?} : ?_{\delta} \sqsubseteq ?_{\delta'}} \quad \frac{\mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2 \quad \mathbf{c}' : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2}{\mathbf{c} \rightarrow \mathbf{c}' : \mathbf{G}_1 \rightarrow \mathbf{G}'_1 \sqsubseteq \mathbf{G}_2 \rightarrow \mathbf{G}'_2} \\
\frac{\mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2 \quad \mathbf{c}' : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2}{\mathbf{c} \times \mathbf{c}' : \mathbf{G}_1 \times \mathbf{G}'_1 \sqsubseteq \mathbf{G}_2 \times \mathbf{G}'_2} \quad \frac{\mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2}{\forall X. \mathbf{c} : \forall X. \mathbf{G}_1 \sqsubseteq \forall X. \mathbf{G}_2} \quad \frac{X : F \in \delta \quad \delta \vdash F}{\text{inj}_X : F \sqsubseteq ?_{\delta}} \\
\frac{}{\text{inj}_B : B \sqsubseteq ?_{\delta}} \quad \frac{\mathbf{c} : \mathbf{G} \sqsubseteq ?_{\delta} \rightarrow ?_{\delta}}{\text{inj}_{\rightarrow}(\mathbf{c}) : \mathbf{G} \sqsubseteq ?_{\delta}} \quad \frac{\mathbf{c} : \mathbf{G} \sqsubseteq ?_{\delta} \times ?_{\delta}}{\text{inj}_{\times}(\mathbf{c}) : \mathbf{G} \sqsubseteq ?_{\delta}} \quad \frac{\mathbf{c} : \mathbf{G} \sqsubseteq \forall X. ?_{\delta, X}}{\text{inj}_{\forall}(\mathbf{c}) : \mathbf{G} \sqsubseteq ?_{\delta}}
\end{array}$$

$\boxed{\mathbf{c}; \mathbf{c} = \mathbf{c}}$ Composition of precision proof terms

$$\begin{array}{l}
\text{refl}_B; \text{refl}_B = \text{refl}_B \quad \text{refl}_X; \text{refl}_X = \text{refl}_X \quad (\mathbf{c}_1 \rightarrow \mathbf{c}_2); (\mathbf{c}'_1 \rightarrow \mathbf{c}'_2) = (\mathbf{c}_1; \mathbf{c}'_1) \rightarrow (\mathbf{c}_2; \mathbf{c}'_2) \\
(\mathbf{c}_1 \times \mathbf{c}_2); (\mathbf{c}'_1 \times \mathbf{c}'_2) = (\mathbf{c}_1; \mathbf{c}'_1) \times (\mathbf{c}_2; \mathbf{c}'_2) \quad (\forall X. \mathbf{c}); (\forall X. \mathbf{c}') = \forall X. (\mathbf{c}; \mathbf{c}') \quad \text{refl}_B; \text{inj}_B = \text{inj}_B \\
\text{refl}_F; \text{inj}_X = \text{inj}_X \quad \mathbf{c}_1; \text{inj}_{\rightarrow}(\mathbf{c}_2) = \text{inj}_{\rightarrow}(\mathbf{c}_1; \mathbf{c}_2) \quad \mathbf{c}; \text{inj}_{?} = \mathbf{c}
\end{array}$$

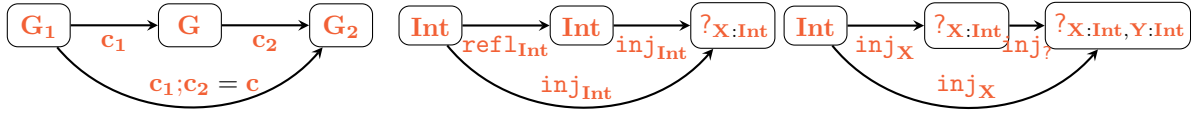


Figure 6.2: Proof-relevant type precision, composition, and examples.

precise than unknown: if \mathbf{c} witnesses that a type \mathbf{G} is more precise than $?_{\delta} \rightarrow ?_{\delta}$, then $\text{inj}_{\rightarrow}(\mathbf{c})$ justifies that \mathbf{G} is more precise than $?_{\delta}$. Similarly, $\text{inj}_{\times}(\mathbf{c})$, and $\text{inj}_{\forall}(\mathbf{c})$ witnesses that pair types and polymorphic types are more precise than unknown respectively. $\text{inj}_{?}$ justifies that an unknown type is more precise than another if the environment of the former is contained in the environment of the latter.

To support transitive judgments of precision, we define the composition of proof terms in Figure 6.2. A reflexive proof term combined with itself yields the same proof term. The combinations of function, pair, and abstraction proof terms are defined inductively. The combination of a reflexive base type proof term with an injection from that type yields the latter. Similarly, the combination of a reflexive refl_X proof term with an injection from X , yields just the injection from X . Finally, the combination of an $\text{inj}_{?}$ from the right can always be dropped. Figure 6.2 illustrates graphically the composition function along some examples. If $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}$, and $\mathbf{c} : \mathbf{G} \sqsubseteq \mathbf{G}_2$, then $\mathbf{c}_1; \mathbf{c}_2 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$. If $\text{refl}_{\text{Int}} : \text{Int} \sqsubseteq \text{Int}$, and $\text{inj}_{\text{Int}} : \text{Int} \sqsubseteq ?_{X:\text{Int}}$, then as $\text{refl}_{\text{Int}}; \text{inj}_{\text{Int}} = \text{inj}_{\text{Int}}$, then $\text{inj}_{\text{Int}} : \text{Int} \sqsubseteq ?_{X:\text{Int}}$. Finally, if $\text{inj}_X : \text{Int} \sqsubseteq ?_{X:\text{Int}}$, and $\text{inj}_{?} : ?_{X:\text{Int}} \sqsubseteq ?_{X:\text{Int}, Y:\text{Int}}$, then $\text{inj}_X : \text{Int} \sqsubseteq ?_{X:\text{Int}, Y:\text{Int}}$. With these reflexivity and composition operators, gradual types and type precision proof terms can be seen as the objects and morphisms of a category, which will be useful in Section 6.3.2.

6.3.2 Evidence and consistent transitivity

As mentioned before, evidence is defined as a set of justifications, accounting for the multiple possibilities in which two types can be consistent. Using proof-relevant type precision, evidence ε is defined as a non-empty set of spans $\{\mathbf{S}, \dots\}$, where a span \mathbf{S} is a tuple $(\mathbf{G}, \mathbf{c}_1, \mathbf{c}_2)$ such that if $\varepsilon : \mathbf{G}_1 \sim \mathbf{G}_2$, then \mathbf{G} is a common more precise type than \mathbf{G}_1 and \mathbf{G}_2 , and \mathbf{c}_1 and \mathbf{c}_2 justify “how”, respectively.

Definition 6.2 $\varepsilon : \mathbf{G}_1 \sim \mathbf{G}_2$ iff $\forall (\mathbf{G}, \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon, \mathbf{c}_1 : \mathbf{G} \sqsubseteq \mathbf{G}_1 \wedge \mathbf{c}_2 : \mathbf{G} \sqsubseteq \mathbf{G}_2$.

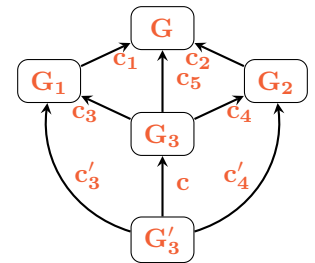
For example, $\varepsilon = \{(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{inj}_{\mathbf{Int}}), (\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{inj}_{\mathbf{X}})\}$ justifies that $\mathbf{Int} \sim ?_{\mathbf{X}:\mathbf{Int}}$, because $\mathbf{refl}_{\mathbf{Int}} : \mathbf{Int} \sqsubseteq \mathbf{Int}$, $\mathbf{inj}_{\mathbf{Int}} : \mathbf{Int} \sqsubseteq ?_{\mathbf{X}:\mathbf{Int}}$, and $\mathbf{inj}_{\mathbf{X}} : \mathbf{Int} \sqsubseteq ?_{\mathbf{X}:\mathbf{Int}}$. Therefore, as explained in Section 6.1, the term $\varepsilon 42 :: ?_{\mathbf{X}:\mathbf{Int}}$ corresponds exactly to the maybe-sealed value $\mathit{maybeSeal}_{\mathbf{X}}(42)$ from the introduction: the evidence holds *both* possible justifications.

The type substitution definition on evidence, and more precisely on proof terms, is fundamental for the plausible sealing mechanism to preserve parametricity. Type substitution for evidence is defined as the type substitution for each of its spans. Type substitution for a span is defined as the type substitution of its components. For example, $(\mathbf{X}, \mathbf{refl}_{\mathbf{X}}, \mathbf{refl}_{\mathbf{X}})[\mathbf{F}/\mathbf{X}] = (\mathbf{F}, \mathbf{refl}_{\mathbf{F}}, \mathbf{refl}_{\mathbf{F}})$ and $(\mathbf{X}, \mathbf{inj}_{\mathbf{X}}, \mathbf{refl}_{\mathbf{X}})[\mathbf{F}/\mathbf{X}] = (\mathbf{F}, \mathbf{inj}_{\mathbf{X}}, \mathbf{refl}_{\mathbf{F}})$. Note that $\mathbf{inj}_{\mathbf{X}}[\mathbf{F}/\mathbf{X}] = \mathbf{inj}_{\mathbf{X}}$ is essential to preserve sealed values; otherwise, we would forget the sealing if we apply the substitution.

Evidence in $\text{GSF}\varepsilon$ and $\text{F}_\varepsilon^?$ have similarities and differences. As we saw earlier (Section 3.5.2), $\text{GSF}\varepsilon$ defines an evidence ε as a pair of evidence types $\langle E_1, E_2 \rangle$, where the consistency judgment is relative to a store ($\varepsilon \Vdash \Xi \vdash G_1 \sim G_2$) and the evidence types satisfy that $\mathit{unlift}(E_1) \sqsubseteq G_1$ and $\mathit{unlift}(E_2) \sqsubseteq G_2$. On the other hand, a $\text{F}_\varepsilon^?$ evidence is a set of spans, where the consistent judgment does not need a store because all the information is stored locally in the unknown types ($\varepsilon : \mathbf{G}_1 \sim \mathbf{G}_2$), and each span $(\mathbf{G}, \mathbf{c}_1, \mathbf{c}_2)$ justifies the same judgment. A $\text{GSF}\varepsilon$ evidence is more comparable to a $\text{F}_\varepsilon^?$ span. For example, evidence $\langle \mathbf{Int}, \alpha^{\mathbf{Int}} \rangle$ and the span $(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{inj}_{\mathbf{X}})$ are both to seal values, and $\langle \alpha^{\mathbf{Int}}, \mathbf{Int} \rangle$ and $(\mathbf{Int}, \mathbf{inj}_{\mathbf{X}}, \mathbf{refl}_{\mathbf{Int}})$ for unsealing. Thus evidence types E and proof terms \mathbf{c} have similar functions, but evidence types are gradual types where type names store type instantiation information, while proof terms precisely describe how one type is more precise than another. For instance, evidence $\langle \mathbf{Int}, \alpha^{\mathbf{Int}} \rangle$ can justify several judgment such as $\mathbf{Int} \sim \alpha$, $\mathbf{Int} \sim ?$, $? \sim \alpha$ and $? \sim ?$. However, the span $(\mathbf{Int}, \mathbf{inj}_{\mathbf{X}}, \mathbf{refl}_{\mathbf{Int}})$ has to belong to an evidence justifying that $\mathbf{Int} \sim ?_{\delta, \mathbf{X}:\mathbf{Int}}$, for some δ .

To define consistent transitivity for this representation of evidence, we first define the *pullback* operator between proof terms.

Lemma 6.3 (Pullback operator and its universal property) *There exists a partial pullback operator such that if $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}$ and $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}$, and $\mathit{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$, then $\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1$, $\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2$, $\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5$ and $\mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}_5$. The pullback operator is universal in the following sense. If there exists \mathbf{G}'_3 , \mathbf{c}'_3 , \mathbf{c}'_4 and \mathbf{c}'_5 such that $\mathbf{c}'_3; \mathbf{c}_1 = \mathbf{c}'_5$ and $\mathbf{c}'_4; \mathbf{c}_2 = \mathbf{c}'_5$, then $\mathit{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$ and there exists a unique $\mathbf{c} : \mathbf{G}'_3 \sqsubseteq \mathbf{G}_3$ such that $\mathbf{c}; \mathbf{c}_3 = \mathbf{c}'_3$, $\mathbf{c}; \mathbf{c}_4 = \mathbf{c}'_4$ and $\mathbf{c}; \mathbf{c}_5 = \mathbf{c}'_5$.*



Our pullback operator and its universal property are a mild adaptation of the standard def-

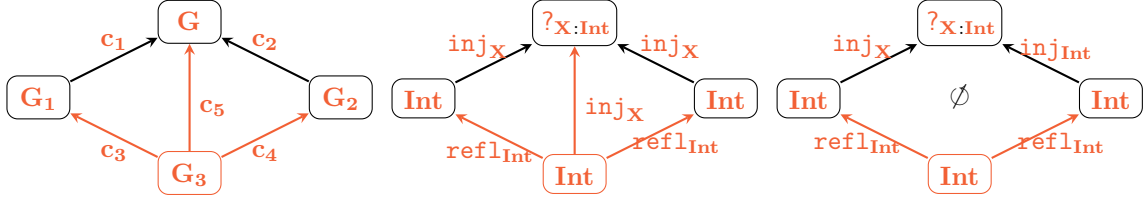


Figure 6.3: Pullback and examples.

initiation in category theory (nLab contributors, 2021a) since the pullback operator is partial, i.e., it is not always defined. Figure 6.3 illustrates our *pullback* operator, along with two examples. The first example (second diagram) calculates $\text{pullback}(\text{?X:Int}, (\text{Int}, \text{inj}_X), (\text{Int}, \text{inj}_X))$, returning $(\text{Int}, \text{refl}_{\text{Int}}, \text{refl}_{\text{Int}}, \text{inj}_X)$. Note that the diamond diagram commutes, obtaining inj_X both on the left and on the right. The second example (third diagram) tries to calculate $\text{pullback}(\text{?X:Int}, (\text{Int}, \text{inj}_X), (\text{Int}, \text{inj}_{\text{Int}}))$, but it is undefined since there is no gradual type and proof terms such that the diagram commutes. The definition of the pullback operator is algorithmic, proceeding in most cases congruently and can be found in Appendix.

The universal property in Lemma 6.3 establishes that if \mathbf{G}'_3 (with proof terms \mathbf{c}'_3 and \mathbf{c}'_4) makes the pullback diagram commute, then the pullback is defined, resulting in type \mathbf{G}_3 (with proof terms \mathbf{c}_3 and \mathbf{c}_4), the less precise type that makes the diagram commute. Additionally, there exists a proof term \mathbf{c} such that $\mathbf{c} : \mathbf{G}_3 \sqsubseteq \mathbf{G}'_3$ and all sub-diagrams commute. For example, suppose $\mathbf{c}_1 = \mathbf{c}_2 = \text{inj}_?$ and $\mathbf{G} = \mathbf{G}_1 = \mathbf{G}_2 = \text{?X:Int}$. Since $\mathbf{G}'_3 = \text{Int}$, with $\mathbf{c}'_3 = \mathbf{c}'_4 = \text{inj}_X$, makes the diagram commute, we know that the pullback exists. In this case, we know that $\text{pullback}(\text{?X:Int}, (\text{?X:Int}, \text{inj}_?), (\text{?X:Int}, \text{inj}_?)) = (\text{?X:Int}, \text{inj}_?, \text{inj}_?, \text{inj}_?)$. Observe that there exist two proof terms \mathbf{c} such that $\mathbf{c} : \mathbf{G}_3 \sqsubseteq \mathbf{G}'_3$, $\mathbf{c} = \text{inj}_{\text{Int}}$ and $\mathbf{c} = \text{inj}_X$. However, only $\mathbf{c} = \text{inj}_X$ satisfies $\mathbf{c};\mathbf{c}_3 = \mathbf{c}'_3$, $\mathbf{c};\mathbf{c}_4 = \mathbf{c}'_4$ and $\mathbf{c};\mathbf{c}_5 = \mathbf{c}'_5$. Consistent transitivity between spans is then defined as follows:

Definition 6.4 (Consistent transitivity for spans) *Let $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}$, $\mathbf{c}'_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}'_1$, $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}$ and $\mathbf{c}'_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}'_2$. We pose $(\mathbf{G}_1, \mathbf{c}'_1, \mathbf{c}_1) \ddagger (\mathbf{G}_2, \mathbf{c}_2, \mathbf{c}'_2) = \{(\mathbf{G}_3, \mathbf{c}_6, \mathbf{c}_7) \mid \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \wedge \mathbf{c}_3;\mathbf{c}'_1 = \mathbf{c}_6 \wedge \mathbf{c}_4;\mathbf{c}'_2 = \mathbf{c}_7\}$.*

The definition is very close to the standard definition of composition of spans (nLab contributors, 2021b), except that we are dealing with a *partial* pullback and a *set of spans* rather than a single span. Figure 6.4 graphically supports the definition of consistent transitivity, along with an example. First the pullback of \mathbf{c}_1 and \mathbf{c}_2 is computed. If the pullback is defined, then the new evidence type is computed using the common gradual type from the pullback \mathbf{G}_3 , and new proofs that \mathbf{G}_3 is more precise than \mathbf{G}'_1 and \mathbf{G}'_2 using the proof-relevant composition operator. Note that the result of consistent transitivity for spans is either a singleton set or the empty set. In the example, we have that $(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_X) : \text{Int} \sim \text{?X:Int}$ (representing a seal at type \mathbf{X}), and $(\text{Int}, \text{inj}_X, \text{inj}_Y) : \text{?X:Int} \sim \text{?Y:Int}$ (an unseal at type \mathbf{X} , followed by a seal at type \mathbf{Y}). Consistent transitivity $(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_X) \ddagger (\text{Int}, \text{inj}_X, \text{inj}_Y)$ is computed by first computing $\text{pullback}(\text{?X:Int}, (\text{Int}, \text{inj}_X), (\text{Int}, \text{inj}_X)) = (\text{Int}, \text{refl}_{\text{Int}}, \text{refl}_{\text{Int}}, \text{inj}_X)$. As $\text{refl}_{\text{Int}};\text{refl}_{\text{Int}} = \text{refl}_{\text{Int}}$, and $\text{refl}_{\text{Int}};\text{inj}_Y = \text{inj}_Y$, the result is $(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_Y)$.

Finally, consistent transitivity between evidences is just defined as the natural lifting of consistent transitivity of spans to sets of spans.

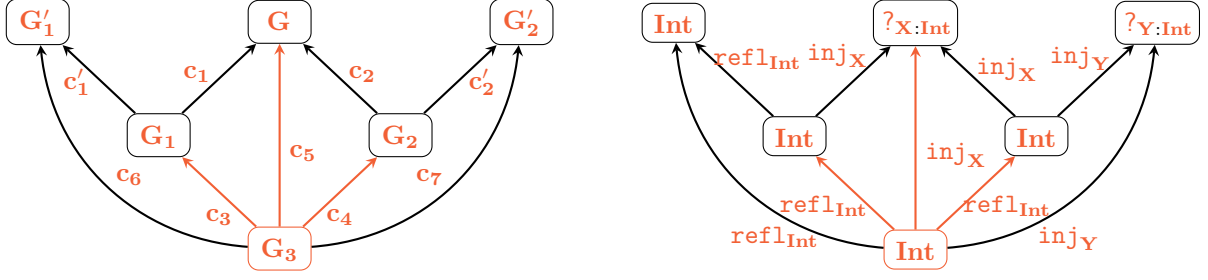


Figure 6.4: Consistent transitivity for spans and example.

Definition 6.5 (Consistent transitivity for evidence) *Let $\varepsilon_1 : G_1 \sim G$, and $\varepsilon_2 : G \sim G_2$.*

$$\varepsilon_1 \circledast \varepsilon_2 ::= \begin{cases} \varepsilon & \text{if } \varepsilon = \{S \mid S \in S_1 \circledast S_2, S_1 \in \varepsilon_1, S_2 \in \varepsilon_2\} \neq \emptyset \\ \text{error} & \text{otherwise} \end{cases}$$

Note that if the resulting set is empty, then consistent transitivity is undefined, representing a runtime type error because plausibility of well-typedness has been refuted. Otherwise, if consistent transitivity is defined, then the obtained evidence justifies the transitive judgment.

Lemma 6.6 *Let $\varepsilon_1 : G_1 \sim G$, and $\varepsilon_2 : G \sim G_2$. If $\varepsilon_1 \circledast \varepsilon_2$ is defined, then $\varepsilon_1 \circledast \varepsilon_2 : G_1 \sim G_2$.*

Associativity of consistent transitivity. Associativity of consistent transitivity is a key property in evidence-based semantics, used to establish type soundness as well as space efficiency optimizations (Toro and Tanter, 2020; Bañados Schwerter et al., 2021). In particular, in this work the associativity lemma is used extensively in the proof of parametricity of $F_\varepsilon^?$ (Section 6.4).

Lemma 6.7 (\checkmark Associativity of Evidence Composition) $(\varepsilon_1 \circledast \varepsilon_2) \circledast \varepsilon_3 = \varepsilon_1 \circledast (\varepsilon_2 \circledast \varepsilon_3)$.

The proof of the associativity lemma relies on the universal property of the pullback (Lemma 6.3).

Examples of reduction. Armed with the dynamic semantics of $F_\varepsilon^?$ and the concrete representation of evidence, we first illustrate the reduction of Example (1) from Section 6.1.2: $(\lambda X. \lambda x. ?x :: X) [\text{Int}] 42$, which reduces to 42. Its elaboration (omitting some trivial evidence for conciseness) and reduction proceed as follows:

$$\begin{aligned}
& (\varepsilon_2 (\Lambda X. \lambda x : ?_X. \varepsilon_1 x :: X [\text{Int}]) :: ? \rightarrow \text{Int}) (\varepsilon_3 42 :: ?) \quad \text{where} \\
& \varepsilon_1 = \{(X, \text{inj}_X, \text{refl}_X)\} \text{ and} \\
& \varepsilon_2 = \{(\text{Int} \rightarrow \text{Int}, \text{inj}_X \rightarrow \text{refl}_{\text{Int}}, \text{inj}_{\text{Int}} \rightarrow \text{refl}_{\text{Int}}), \\
& \quad (? \rightarrow \text{Int}, \text{inj}_? \rightarrow \text{refl}_{\text{Int}}, \text{inj}_? \rightarrow \text{refl}_{\text{Int}})\} \text{ and} \\
& \varepsilon_3 = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\text{Int}})\} \\
(\text{RappG}) \mapsto & (\varepsilon_2 (\lambda x : ?_X : \text{Int}. \varepsilon'_1 x :: \text{Int}) :: ? \rightarrow \text{Int}) (\varepsilon_3 42 :: ?) \quad \text{where} \\
& \varepsilon'_1 = \{(\text{Int}, \text{inj}_X, \text{refl}_{\text{Int}})\} \\
(\text{Rapp}) \mapsto & \text{cod}(\varepsilon_2) (\varepsilon'_1 (\varepsilon'_3 42 :: ?) :: \text{Int}) :: \text{Int} \quad \text{where} \\
& \varepsilon'_3 = \varepsilon_3 \circ \text{dom}(\varepsilon_2) = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_X), (\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\text{Int}})\} \\
(\text{Rasc}) \mapsto & \text{cod}(\varepsilon_2) (\varepsilon_4 42 :: \text{Int}) :: \text{Int} \quad \text{where} \\
& \varepsilon_4 = \varepsilon'_3 \circ \varepsilon'_1 = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{refl}_{\text{Int}})\} \\
(\text{Rasc}) \mapsto & \varepsilon_5 42 :: \text{Int} \quad \text{where} \\
& \varepsilon_5 = \varepsilon_4 \circ \text{cod}(\varepsilon_2) = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{refl}_{\text{Int}})\}
\end{aligned}$$

We now illustrate the reduction of Example (3) from Section 6.1.2: $(\Lambda X. \lambda x : ?_x :: X) [\text{Bool}] 42$. The elaboration and reduction are as follows:

$$\begin{aligned}
& (\varepsilon_2 (\Lambda X. \lambda x : ?_X. \varepsilon_1 x :: X [\text{Bool}]) :: ? \rightarrow \text{Bool}) (\varepsilon_3 42 :: ?) \quad \text{where} \\
& \varepsilon_1 = \{(X, \text{inj}_X, \text{refl}_X)\} \text{ and} \\
& \varepsilon_2 = \{(\text{Bool} \rightarrow \text{Bool}, \text{inj}_X \rightarrow \text{refl}_{\text{Bool}}, \text{inj}_{\text{Bool}} \rightarrow \text{refl}_{\text{Bool}}), \\
& \quad (? \rightarrow \text{Bool}, \text{inj}_? \rightarrow \text{refl}_{\text{Bool}}, \text{inj}_? \rightarrow \text{refl}_{\text{Bool}})\} \text{ and} \\
& \varepsilon_3 = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\text{Int}})\} \\
(\text{RappG}) \mapsto & (\varepsilon_2 (\lambda x : ?_X : \text{Bool}. \varepsilon'_1 x :: \text{Bool}) :: ? \rightarrow \text{Bool}) (\varepsilon_3 42 :: ?) \quad \text{where} \\
& \varepsilon'_1 = \{(\text{Bool}, \text{inj}_X, \text{refl}_{\text{Bool}})\} \\
(\text{Rapp}) \mapsto & \text{cod}(\varepsilon_2) (\varepsilon'_1 (\varepsilon'_3 42 :: ?) :: \text{Bool}) :: \text{Bool} \quad \text{where} \\
& \varepsilon'_3 = \varepsilon_3 \circ \text{dom}(\varepsilon_2) = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\text{Int}})\} \\
(\text{Rasc}) \mapsto & \text{error} \quad \text{because} \\
& \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\text{Int}})\} \circ \{(\text{Bool}, \text{inj}_X, \text{refl}_{\text{Bool}})\} \text{ is undefined}
\end{aligned}$$

Finally, we show the reduction of Example (5) from Section 6.1.2, which illustrates the prevention of a violation of parametricity at runtime: $(\Lambda X. \lambda x : X. (x :: ?) + 1) [\text{Int}] 3$. The elaboration and reduction are as follows:

$$\begin{aligned}
& (\varepsilon_3 ((\Lambda X. \lambda x : X. (\varepsilon_2 (\varepsilon_1 x :: ?_X) :: \text{Int}) + (\varepsilon_{\text{Int}} 1 :: \text{Int})) [\text{Int}]) :: \text{Int} \rightarrow \text{Int}) (\varepsilon_{\text{Int}} 3 :: \text{Int}) \\
& \varepsilon_1 = \{(X, \text{refl}_X, \text{inj}_X)\} \text{ and} \\
& \varepsilon_2 = \{(\text{Int}, \text{inj}_{\text{Int}}, \text{refl}_{\text{Int}})\} \text{ and} \\
& \varepsilon_{\text{Int}} = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{refl}_{\text{Int}})\} \text{ and} \\
& \varepsilon_3 = \{(\text{Int} \rightarrow \text{Int}, \text{refl}_{\text{Int}} \rightarrow \text{refl}_{\text{Int}}, \text{refl}_{\text{Int}} \rightarrow \text{refl}_{\text{Int}})\} \\
(\text{RappG}) \mapsto & (\varepsilon_3 (\lambda x : \text{Int}. (\varepsilon_2 (\varepsilon'_1 x :: ?_{X:\text{Int}}) :: \text{Int}) + (\varepsilon_{\text{Int}} 1 :: \text{Int})) :: \text{Int} \rightarrow \text{Int}) (\varepsilon_{\text{Int}} 3 :: \text{Int}) \\
& \text{where } \varepsilon'_1 = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_X)\} \\
(\text{Rapp}) \mapsto & \text{cod}(\varepsilon_3) ((\varepsilon_2 (\varepsilon'_1 (\varepsilon_{\text{Int}} 3 :: \text{Int}) :: ?_{X:\text{Int}}) :: \text{Int}) + (\varepsilon_{\text{Int}} 1 :: \text{Int})) :: \text{Int} \\
& \text{where } \varepsilon_{\text{Int}} \circ \text{dom}(\varepsilon_3) = \varepsilon_{\text{Int}} \\
(\text{Rasc}) \mapsto & \text{cod}(\varepsilon_3) ((\varepsilon_2 (\varepsilon'_1 3 :: ?_{X:\text{Int}}) :: \text{Int}) + (\varepsilon_{\text{Int}} 1 :: \text{Int})) :: \text{Int} \\
& \text{where } \varepsilon_{\text{Int}} \circ \varepsilon'_1 = \varepsilon'_1 \\
(\text{Rasc}) \mapsto & \text{error} \\
& \text{because } \varepsilon'_1 \circ \varepsilon_2 = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_X)\} \circ \{(\text{Int}, \text{inj}_{\text{Int}}, \text{refl}_{\text{Int}})\} \text{ is undefined}
\end{aligned}$$

6.4 $F_\varepsilon^?$: Gradual Parametricity

In this section, we present parametricity for $F_\varepsilon^?$. We use a standard technique for establishing this result: *step-indexed* logical relations (Appel and McAllester, 2001; Ahmed, 2006). Step indexing ensures the well-foundedness of the logical relation. We start by defining the logical relation for values and terms, and then we establish the fundamental property or parametricity. Our proposal is the first gradual polymorphic language to support a formulation of parametricity where semantic types are tracked in a lexical environment, similar to traditional formulations of parametricity (Reynolds, 1983).

Logical relations. Figure 6.5 presents the logical relation for parametricity along with some auxiliary definitions. The relational interpretation is presented using *atoms* of the form $(n, \mathbf{t}_1, \mathbf{t}_2) \in \text{ATOM}[\mathbf{G}_1, \mathbf{G}_2]$, where n denotes the step index, and \mathbf{t}_1 and \mathbf{t}_2 denote closed well-typed terms at types \mathbf{G}_1 and \mathbf{G}_2 , respectively. The logical relation is defined using two mutually-defined interpretations: one for values $\mathcal{V}_\rho[\mathbf{G}]$ and one for computations $\mathcal{T}_\rho[\mathbf{G}]$. Both interpretations are indexed by a type \mathbf{G} , and an environment ρ , which maps type variables to two types \mathbf{G}_1 and \mathbf{G}_2 and a relation $R \in \text{REL}[\mathbf{G}_1, \mathbf{G}_2]$. $\text{REL}[\mathbf{G}_1, \mathbf{G}_2]$ defines the set of all admissible relations R such that $R \subseteq \text{ATOM}^{\text{val}}[\mathbf{G}_1, \mathbf{G}_2]$ (the subset of atoms where terms are values). For convenience, if $\rho = \{\overline{\mathbf{X}_i} \mapsto (\mathbf{G}_{i1}, \mathbf{G}_{i2}, R_i)\}$, then $\rho.1$, $\rho.2$, and $\rho.R$ are abbreviations for $\{\overline{\mathbf{X}_i} \mapsto \mathbf{G}_{i1}\}$ and $\{\overline{\mathbf{X}_i} \mapsto \mathbf{G}_{i2}\}$, and $\{\overline{\mathbf{X}_i} \mapsto R_i\}$ respectively. Thus $\rho.j(\mathbf{G})$ is an abbreviation for multiple substitutions $\mathbf{G}[\overline{\mathbf{X}_i} \mapsto \mathbf{G}_{ij}]$. Finally, $\text{ATOM}_\rho[\mathbf{G}]$ denotes the set of atoms $\text{ATOM}[\rho.1(\mathbf{G}), \rho.2(\mathbf{G})]$.

Logical relation for values. The definition of related values is standard except for the unknown type. Two base values of type \mathbf{B} are related if they are the same. Two functions are related at type $\mathbf{G}_1 \rightarrow \mathbf{G}_2$, if given two related arguments at type \mathbf{G}_1 (and a strictly smaller index), the application yields related computations at type \mathbf{G}_2 . We use notation $\triangleright^i(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$ as an abbreviation for $(n - i, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$, and $\triangleright(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$ for $\triangleright^1(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$. Two pairs are related at type $\mathbf{G}_1 \times \mathbf{G}_2$ if the first and second projections are related at types \mathbf{G}_1 and \mathbf{G}_2 respectively. Two type abstractions are related if their instantiations to two arbitrary base types yields related computations for any given relation between the instantiated types. Two values are related at an abstract type \mathbf{X} , if they are contained in the relation for \mathbf{X} .

Two values are related at the unknown type $?_\delta$, if given any evidence ε that justifies that any \mathbf{G}_R is ground with respect to δ , notation $\vdash \varepsilon : \delta \rightarrow \mathbf{G}_R$, then both values ascribed to $\rho.1(\mathbf{G}_R)$ and $\rho.2(\mathbf{G}_R)$, using $\rho.1(\varepsilon)$ and $\rho.2(\varepsilon)$ respectively, are related computations at type \mathbf{G}_R . This definition captures the fact that if two values are related at $?_\delta$, they are also related at some more precise ground type, either \mathbf{X} , \mathbf{B} , $?_\delta \times ?_\delta$, $?_\delta \rightarrow ?_\delta$, or $\forall \mathbf{X}. ?_\delta, \mathbf{X}$, after removing the respective injections to the unknown type with the evidences $\rho.1(\varepsilon)$ and $\rho.2(\varepsilon)$. Both λB and GSF use similar approaches for defining the logical relation for values of type unknown but are formalized differently, according to the syntax of the considered languages. Relation $\vdash \varepsilon : \delta \rightarrow \mathbf{G}_R$ is defined such that \mathbf{G}_R is a *ground type* restricted to δ , and $\varepsilon : ?_\delta \sim \mathbf{G}_R$:

$$\begin{array}{c}
 \hline
 \vdash \{(\mathbf{B}, \text{inj}_{\mathbf{B}}, \text{refl}_{\mathbf{B}})\} : \delta \rightarrow \mathbf{B} \\
 \hline
 \mathbf{X} : \mathbf{F} \in \delta \quad \delta \vdash \mathbf{F} \\
 \hline
 \vdash \{(\mathbf{F}, \text{inj}_{\mathbf{X}}, \text{refl}_{\mathbf{F}})\} : \delta \rightarrow \mathbf{F} \\
 \hline
 \end{array}
 \qquad
 \begin{array}{c}
 \hline
 \vdash \{(?_\delta \rightarrow ?_\delta, \text{inj}_{\rightarrow}, \text{inj}_{? \rightarrow ?})\} : \delta \rightarrow ?_\delta \rightarrow ?_\delta \\
 \hline
 \vdash \{(?_\delta \times ?_\delta, \text{inj}_{\times}, \text{inj}_{? \times ?})\} : \delta \rightarrow ?_\delta \times ?_\delta \\
 \hline
 \vdash \{(\forall \mathbf{X}. ?_\delta, \mathbf{X}, \text{inj}_{\forall}, \forall \mathbf{X}. \text{inj}_{?})\} : \delta \rightarrow \forall \mathbf{X}. ?_\delta, \mathbf{X} \\
 \hline
 \end{array}$$

$\mathcal{V}_\rho[B]$	$= \{(n, \mathbf{v}, \mathbf{v}) \in \text{ATOM}_\rho[B]\}$
$\mathcal{V}_\rho[\mathbf{G}_1 \rightarrow \mathbf{G}_2]$	$= \{(n, \mathbf{v}_1, \mathbf{v}_2) \in \text{ATOM}_\rho[\mathbf{G}_1 \rightarrow \mathbf{G}_2] \mid \forall n' \leq n, \mathbf{v}'_1, \mathbf{v}'_2. \\ \triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[\mathbf{G}_1] \Rightarrow (n', \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{T}_\rho[\mathbf{G}_2]\}$
$\mathcal{V}_\rho[\mathbf{G}_1 \times \mathbf{G}_2]$	$= \{(n, \mathbf{v}_1, \mathbf{v}_2) \in \text{ATOM}_\rho[\mathbf{G}_1 \times \mathbf{G}_2] \mid (n, \pi_1(\mathbf{v}_1), \pi_1(\mathbf{v}_2)) \in \mathcal{T}_\rho[\mathbf{G}_1] \wedge \\ (n, \pi_2(\mathbf{v}_1), \pi_2(\mathbf{v}_2)) \in \mathcal{T}_\rho[\mathbf{G}_2]\}$
$\mathcal{V}_\rho[\forall \mathbf{X}. \mathbf{G}]$	$= \{(n, \mathbf{v}_1, \mathbf{v}_2) \in \text{ATOM}_\rho[\forall \mathbf{X}. \mathbf{G}] \mid \forall \vdash \mathbf{B}_1, \vdash \mathbf{B}_2, R \in \text{REL}[\mathbf{B}_1, \mathbf{B}_2]. \\ (n, \mathbf{v}_1 [\mathbf{B}_1], \mathbf{v}_2 [\mathbf{B}_2]) \in \mathcal{T}_{\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R)}[\mathbf{G}]\}$
$\mathcal{V}_\rho[\mathbf{X}]$	$= \rho.R(\mathbf{X})$
$\mathcal{V}_\rho[?_\delta]$	$= \{(n, \mathbf{v}_1, \mathbf{v}_2) \in \text{ATOM}_\rho[?_\delta] \mid \forall \mathbf{G}_R, \varepsilon, \vdash \varepsilon : \delta \rightarrow \mathbf{G}_R. \\ (n, \rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}_R), \rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}_R)) \in \mathcal{T}_\rho[\mathbf{G}_R]\}$
$\mathcal{T}_\rho[\mathbf{G}]$	$= \{(n, \mathbf{t}_1, \mathbf{t}_2) \in \text{ATOM}_\rho[\mathbf{G}] \mid \forall i < n. \\ (\forall \mathbf{v}_1. \mathbf{t}_1 \mapsto^i \mathbf{v}_1 \Rightarrow \exists \mathbf{v}_2. \mathbf{t}_2 \mapsto^* \mathbf{v}_2 \wedge \triangleright^i (n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]) \wedge \\ (\mathbf{t}_1 \mapsto^i \mathbf{error} \Rightarrow \mathbf{t}_2 \mapsto^* \mathbf{error})\}$
$\mathcal{D}[\cdot]$	$= \{(n, \emptyset)\}$
$\mathcal{D}[\Delta, \mathbf{X}]$	$= \{(n, \rho[\mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R)]) \mid (n, \rho) \in \mathcal{D}[\Delta] \wedge R \in \text{REL}[\mathbf{B}_1, \mathbf{B}_2]\}$
$\mathcal{G}_\rho[\cdot]$	$= \{(n, \emptyset)\}$
$\mathcal{G}_\rho[\Gamma, \mathbf{x} : \mathbf{G}]$	$= \{(n, \gamma[\mathbf{x} \mapsto (\mathbf{v}_1, \mathbf{v}_2)]) \mid (n, \gamma) \in \mathcal{G}_\rho[\Gamma] \wedge (n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]\}$
$\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G} \triangleq \Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G} \wedge \Delta; \Gamma \vdash \mathbf{t}_2 : \mathbf{G} \wedge \forall n, \rho, \gamma. ((n, \rho) \in \mathcal{D}[\Delta] \wedge (n, \gamma) \in \mathcal{G}_\rho[\Gamma]) \Rightarrow \\ (n, \rho_1(\gamma_1(\mathbf{t}_1)), \rho_2(\gamma_2(\mathbf{t}_2))) \in \mathcal{T}_\rho[\mathbf{G}]$	
$\Delta; \Gamma \vdash \mathbf{t}_1 \approx \mathbf{t}_2 : \mathbf{G} \triangleq \Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G} \wedge \Delta; \Gamma \vdash \mathbf{t}_2 \preceq \mathbf{t}_1 : \mathbf{G}$	

$$\begin{aligned}
\text{ATOM}_\rho[\mathbf{G}] &= \{(n, \mathbf{t}_1, \mathbf{t}_2) \in \text{ATOM}[\rho.1(\mathbf{G}), \rho.2(\mathbf{G})]\} \\
\text{ATOM}[\mathbf{G}_1, \mathbf{G}_2] &= \{(n, \mathbf{t}_1, \mathbf{t}_2) \mid \vdash \mathbf{t}_1 : \mathbf{G}_1 \wedge \vdash \mathbf{t}_2 : \mathbf{G}_2\} \\
\text{ATOM}^{\text{val}}[\mathbf{G}_1, \mathbf{G}_2] &= \{(n, \mathbf{v}_1, \mathbf{v}_2) \in \text{ATOM}[\mathbf{G}_1, \mathbf{G}_2]\} \\
\text{REL}[\mathbf{G}_1, \mathbf{G}_2] &= \{R \subseteq \text{ATOM}^{\text{val}}[\mathbf{G}_1, \mathbf{G}_2] \mid \forall n' \leq n, \mathbf{v}_1, \mathbf{v}_2. (n, \mathbf{v}_1, \mathbf{v}_2) \in R \Rightarrow (n', \mathbf{v}_1, \mathbf{v}_2) \in R\}
\end{aligned}$$

Figure 6.5: Gradual logical relation and auxiliary definitions.

Let us illustrate the interpretation of the unknown type with examples. Consider the evidences:

$$\varepsilon_{\text{Int}^?} = \{(\mathbf{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\text{Int}})\}, \varepsilon_{\text{Int}} = \{(\mathbf{Int}, \text{refl}_{\text{Int}}, \text{refl}_{\text{Int}})\} \text{ and } \varepsilon_{\mathbf{X}^?} = \{(\mathbf{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\mathbf{X}})\}$$

where $\delta = \mathbf{X} : \mathbf{X}$ and $\delta' = \mathbf{X} : \text{Int}$

- $(n, \varepsilon_{\text{Int}^?42} :: ?_{\delta'}, \varepsilon_{\text{Int}^?42} :: ?_{\delta'}) \in \mathcal{V}_\rho[?_\delta]$ because for $\varepsilon = \{(\mathbf{Int}, \text{inj}_{\text{Int}}, \text{refl}_{\text{Int}})\}$ and $\vdash \varepsilon : \delta \rightarrow \mathbf{Int}$, as $\varepsilon_{\text{Int}^?} \dot{\circ} \rho.i(\varepsilon) = \varepsilon_{\text{Int}}$, then $\triangleright (n, \varepsilon_{\text{Int}42} :: \mathbf{Int}, \varepsilon_{\text{Int}42} :: \mathbf{Int}) \in \mathcal{V}_\rho[\mathbf{Int}]$ (and for every other evidence ε and \mathbf{G}_R , such that $\vdash \varepsilon : \delta \rightarrow \mathbf{G}_R$, consistent transitivity is not defined).
- $(n, \varepsilon_{\text{Int}^?42} :: ?_{\delta'}, \varepsilon_{\text{Int}^?43} :: ?_{\delta'}) \notin \mathcal{V}_\rho[?_\delta]$ because for $\vdash \{(\mathbf{Int}, \text{inj}_{\text{Int}}, \text{refl}_{\text{Int}})\} : \delta \rightarrow \mathbf{Int}$, as $\triangleright (n, \varepsilon_{\text{Int}42} :: \mathbf{Int}, \varepsilon_{\text{Int}43} :: \mathbf{Int}) \notin \mathcal{V}_\rho[\mathbf{Int}]$.
- Suppose $\triangleright (n, \varepsilon_{\text{Int}42} :: \mathbf{Int}, \varepsilon_{\text{Int}43} :: \mathbf{Int}) \in \rho.R(\mathbf{X})$. Then $(n, \varepsilon_{\mathbf{X}^?42} :: ?_{\delta'}, \varepsilon_{\mathbf{X}^?43} :: ?_{\delta'}) \in \mathcal{V}_\rho[?_\delta]$ because for $\varepsilon = \{(\mathbf{X}, \text{inj}_{\mathbf{X}}, \text{refl}_{\mathbf{X}})\}$ and $\vdash \varepsilon : \delta \rightarrow \mathbf{X}$, as $\varepsilon_{\mathbf{X}^?} \dot{\circ} \rho.i(\varepsilon) = \varepsilon_{\text{Int}}$, then $\triangleright (n, \varepsilon_{\text{Int}42} :: \mathbf{Int}, \varepsilon_{\text{Int}43} :: \mathbf{Int}) \in \mathcal{V}_\rho[\mathbf{X}] = \rho.R(\mathbf{X})$ (and for every other evidence ε and \mathbf{G}_R , such that $\vdash \varepsilon : \delta \rightarrow \mathbf{G}_R$, consistent transitivity is not defined).
- But $(n, \varepsilon_{\mathbf{X}^?42} :: ?_{\delta'}, \varepsilon_{\mathbf{X}^?43} :: ?_{\delta'}) \notin \mathcal{V}_\rho[?_{\delta'}]$ because for $\varepsilon = \{(\mathbf{Int}, \text{inj}_{\mathbf{X}}, \text{refl}_{\text{Int}})\}$ and $\vdash \varepsilon : \delta' \rightarrow \mathbf{Int}$, as $\varepsilon_{\mathbf{X}^?} \dot{\circ} \rho.i(\varepsilon) = \varepsilon_{\text{Int}}$, but $\triangleright (n, \varepsilon_{\text{Int}42} :: \mathbf{Int}, \varepsilon_{\text{Int}43} :: \mathbf{Int}) \notin \mathcal{V}_\rho[\mathbf{Int}]$.
- Suppose $\rho.R(\mathbf{X}) = \mathcal{V}_\rho[\mathbf{Int}]$, and $\varepsilon_m = \varepsilon_{\text{Int}^?} \cup \varepsilon_{\mathbf{X}^?}$. Then $(n, \varepsilon_m42 :: ?_{\delta'}, \varepsilon_m42 :: ?_{\delta'}) \in \mathcal{V}_\rho[?_\delta]$ because (1) for $\varepsilon = \{(\mathbf{X}, \text{inj}_{\mathbf{X}}, \text{refl}_{\mathbf{X}})\}$ and $\vdash \varepsilon : \delta \rightarrow \mathbf{X}$, as $\varepsilon_m \dot{\circ} \rho.i(\varepsilon) = \varepsilon_{\mathbf{X}^?} \dot{\circ} \rho.i(\varepsilon) = \varepsilon_{\text{Int}}$, then $\triangleright (n, \varepsilon_{\text{Int}42} :: \mathbf{Int}, \varepsilon_{\text{Int}42} :: \mathbf{Int}) \in \mathcal{V}_\rho[\mathbf{X}] = \rho.R(\mathbf{X}) = \mathcal{V}_\rho[\mathbf{Int}]$; and (2) for $\varepsilon = \{(\mathbf{Int}, \text{inj}_{\text{Int}}, \text{refl}_{\text{Int}})\}$, $\vdash \varepsilon : \delta \rightarrow \mathbf{Int}$, as $\varepsilon_m \dot{\circ} \rho.i(\varepsilon) = \varepsilon_{\text{Int}^?} \dot{\circ} \rho.i(\varepsilon) = \varepsilon_{\text{Int}}$, then $\triangleright (n, \varepsilon_{\text{Int}42} :: \mathbf{Int}, \varepsilon_{\text{Int}42} :: \mathbf{Int}) \in \mathcal{V}_\rho[\mathbf{Int}]$.

Note that for the case of functions, pairs, type applications, and the unknown type, although the same step index is used in every recursive reasoning, the relations are well-formed as in each case a single step of reduction is always taken, lowering the index by one.

Logical relation for terms. Two computations are related at n steps if the first term yields a value in $i < n$ reduction steps, then the second must produce a value related at that type at $n - i$ steps; and if the first term fails, then the second also fails.

Logical relation for environments. The interpretation of environment Δ , specifies all type substitutions ρ , such that all type variables in Δ are mapped to a pair of base types and a relation at those types. The interpretation of environment Γ , specifies all value substitution γ , such that every variable of type \mathbf{G} is mapped to a pair of related values at that type.

Parametricity. The logical approximation $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}$ states that given any step index, any environments ρ and γ that satisfy Δ and Γ respectively, the substituted terms are related computations. Similarly to ρ , for convenience if $\gamma = \{\mathbf{x} \mapsto (\mathbf{v}_{i1}, \mathbf{v}_{i2})\}$, then $\gamma_j = \{\mathbf{x} \mapsto \mathbf{v}_{ij}\}$. Finally, the fundamental property states that any well-typed term logically approximates itself.

Theorem 6.8 (Fundamental Property) *If $\Delta; \Gamma \vdash \mathbf{t} : \mathbf{G}$ then $\Delta; \Gamma \vdash \mathbf{t} \preceq \mathbf{t} : \mathbf{G}$.*

As standard (Ahmed, 2004), the proofs of the fundamental property depends on numerous compatibility lemmas for each term constructor and the compositionality lemma, which in this work resembles compositionality for System F.

Lemma 6.9 (Compositionality) *Let $\Delta \vdash \mathbf{F}$, $\Delta, \mathbf{X} \vdash \mathbf{G}$, $(n, \rho) \in \mathcal{D}[\Delta]$, and $R = \mathcal{V}_\rho[\mathbf{F}]$, then $\mathcal{V}_\rho[\mathbf{G}[\mathbf{F}/\mathbf{X}]] = \mathcal{V}_{\rho, \mathbf{X} \mapsto (\rho_1(\mathbf{F}), \rho_2(\mathbf{F}), R)}[\mathbf{G}]$.*

The most important lemma, used by almost all compatibility lemmas and compositionality is the ascription lemma, which says that the ascription of two related values yields related computations.

Lemma 6.10 (Ascription Lemma) *If $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$, $(n, \rho) \in \mathcal{D}[\Delta]$, $\Delta \vdash \mathbf{G}'$ and $\varepsilon : \mathbf{G} \sim \mathbf{G}'$, then $(n, \rho_1(\varepsilon)\mathbf{v}_1 :: \rho_1(\mathbf{G}'), \rho_2(\varepsilon)\mathbf{v}_2 :: \rho_2(\mathbf{G}')) \in \mathcal{T}_\rho[\mathbf{G}']$.*

We finalize this section by emphasizing that most of the logical relations and main lemmas are standard and defined just as in System F. In particular, and contrary to other gradual parametricity formulations, the definition of related values at polymorphic types is defined just as in System F, without the need for special notations and cases. The only unusual case is the definition of related values at the unknown type, but that is expected for any gradual language.

6.5 $\mathbf{F}_\varepsilon^?$: Gradual Guarantees

This section presents graduality for $\mathbf{F}_\varepsilon^?$. We start by presenting the definition of evidence and term precision. Similar to type precision, these definitions are also proof-relevant. Then, we show two of the main challenges of proving graduality: monotonicity of consistent transitivity and monotonicity of type substitution over evidence. We end this section by establishing graduality, more specifically, the static and dynamic gradual guarantees (Siek et al., 2015a).

6.5.1 Evidence Precision

To define precision between evidence we start by stating two intuitive requirements. Suppose $\varepsilon_1 : \mathbf{G}_1 \sim \mathbf{G}'_1$ and $\varepsilon_2 : \mathbf{G}_2 \sim \mathbf{G}'_2$. We say that ε_1 is more precise than ε_2 , if first, the types involved in the judgments are related by precision, *i.e.* $\mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$ and $\mathbf{c}' : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$ for some \mathbf{c} and \mathbf{c}' ; and second, we require that for all $\mathbf{S}_1 \in \varepsilon_1$ there exists some $\mathbf{S}_2 \in \varepsilon_2$, such that \mathbf{S}_1 is more precise than \mathbf{S}_2 . Note that there may be some $\mathbf{S} \in \varepsilon_2$ not in precision with any element of ε_1 . This is intuitively expected by graduality, as it may cause ε_2 to “fail less” than ε_1 when combined with other evidence. Precision between spans $(\mathbf{G}_{t1}, \mathbf{c}_1, \mathbf{c}'_1) \sqsubseteq (\mathbf{G}_{t2}, \mathbf{c}_2, \mathbf{c}'_2)$ could be naively defined if there exists some proof term \mathbf{c}_t that justifies that \mathbf{G}_{t1} is more precise than \mathbf{G}_{t2} , *i.e.* $\mathbf{c}_t : \mathbf{G}_{t1} \sqsubseteq \mathbf{G}_{t2}$.

However, the above requirements are not sufficient to define precision among evidences. Suppose that we have $\varepsilon_1 = \{(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{inj}_{\mathbf{X}})\}$ and $\varepsilon_2 = \{(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{inj}_{\mathbf{Int}})\}$, where $\varepsilon_i : \mathbf{Int} \sim ?_{\mathbf{X}:\mathbf{Int}}$ (Figure 6.6 supports this example). These two evidences meet all the above requirements: there exist $\mathbf{c} = \mathbf{refl}_{\mathbf{Int}}$, $\mathbf{c}' = \mathbf{inj}_{?}$, and $\mathbf{c}_t = \mathbf{refl}_{\mathbf{Int}}$ such that $\mathbf{c} : \mathbf{Int} \sqsubseteq \mathbf{Int}$, $\mathbf{c}' : ?_{\mathbf{X}:\mathbf{Int}} \sqsubseteq ?_{\mathbf{X}:\mathbf{Int}}$ and $\mathbf{c}_t : \mathbf{Int} \sqsubseteq \mathbf{Int}$. We may be tempted to say that $\varepsilon_1 \sqsubseteq \varepsilon_2$ (or vice versa), but then graduality would not hold. In particular, monotonicity of consistent transitivity (MCT), a key lemma used to prove graduality, would be broken. MCT states that given two pairs of evidence related by precision $\varepsilon_1 \sqsubseteq \varepsilon_2$ and $\varepsilon'_1 \sqsubseteq \varepsilon'_2$, if $\varepsilon_1 \mathbin{\text{;}} \varepsilon'_1$ is defined, then $\varepsilon_1 \mathbin{\text{;}} \varepsilon'_1 \sqsubseteq \varepsilon_2 \mathbin{\text{;}} \varepsilon'_2$. In the example, if we take evidence

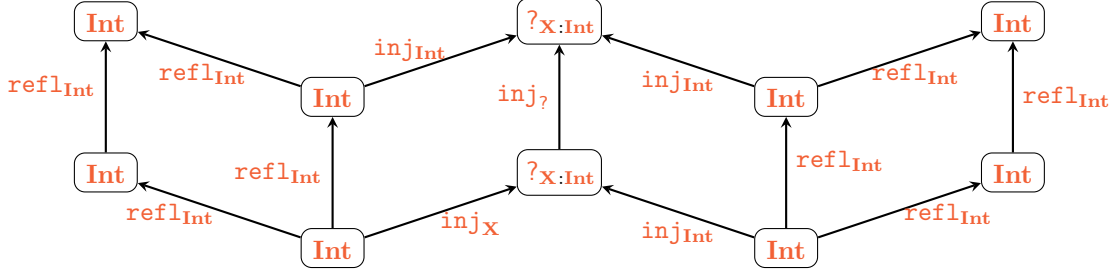
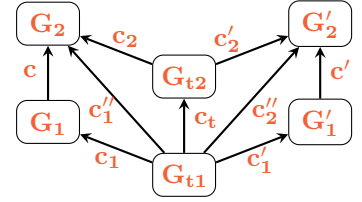


Figure 6.6: Evidence precision auxiliary example.

$\varepsilon'_1 = \varepsilon'_2 = \{(\mathbf{Int}, \mathbf{inj}_{\mathbf{Int}}, \mathbf{refl}_{\mathbf{Int}})\}$ that justifies $?X:\mathbf{Int} \sim \mathbf{Int}$ (and $\varepsilon'_1 \sqsubseteq \varepsilon'_2$), then $\varepsilon_1 \mathbin{\text{\textcircled{;}}} \varepsilon'_1 = \{(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{refl}_{\mathbf{Int}})\}$ is defined but $\varepsilon_2 \mathbin{\text{\textcircled{;}}} \varepsilon'_2$ is not. We can use an analogous reasoning when assuming $\varepsilon_2 \sqsubseteq \varepsilon_1$, using $\varepsilon'_1 = \varepsilon'_2 = \{(\mathbf{Int}, \mathbf{inj}_X, \mathbf{refl}_{\mathbf{Int}})\}$. These two evidences should not be related by precision; we miss a connection between \mathbf{c}_t and both \mathbf{c}' and \mathbf{c} as described next.

Definition 6.11 (Evidence Precision) *If $\varepsilon_1 : \mathbf{G}_1 \sim \mathbf{G}'_1$, $\varepsilon_2 : \mathbf{G}_2 \sim \mathbf{G}'_2$, $\mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$ and $\mathbf{c}' : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$, then we say that $[\mathbf{c}]\varepsilon_1 \sqsubseteq \varepsilon_2[\mathbf{c}']$ iff for all $(\mathbf{G}_{t1}, \mathbf{c}_1, \mathbf{c}'_1) \in \varepsilon_1$ there exists a $(\mathbf{G}_{t2}, \mathbf{c}_2, \mathbf{c}'_2) \in \varepsilon_2$, $\mathbf{c}''_1, \mathbf{c}''_2$ and \mathbf{c}_t such that $\mathbf{c}_t : \mathbf{G}_{t1} \sqsubseteq \mathbf{G}_{t2}$, $\mathbf{c}_t; \mathbf{c}_2 = \mathbf{c}''_1$, $\mathbf{c}_1; \mathbf{c} = \mathbf{c}''_1$, $\mathbf{c}_t; \mathbf{c}'_2 = \mathbf{c}''_2$ and $\mathbf{c}'_1; \mathbf{c}' = \mathbf{c}''_2$.*



In addition to the requirements described above, evidence precision also requires that the combination of \mathbf{c}_t and \mathbf{c}_2 must commute with the combination of \mathbf{c}_1 and \mathbf{c} ; similarly, the combination of \mathbf{c}_t and \mathbf{c}'_2 must commute with the combination of \mathbf{c}'_1 and \mathbf{c}' . Going back to the example, ε_1 and ε_2 are not related by precision as the diagram does not commute: $\mathbf{refl}_{\mathbf{Int}}; \mathbf{inj}_{\mathbf{Int}} = \mathbf{inj}_{\mathbf{Int}}$ ($\mathbf{c}_t; \mathbf{c}'_2 = \mathbf{c}''_2$) and $\mathbf{inj}_X; \mathbf{inj}_? = \mathbf{inj}_X$ ($\mathbf{c}'_1; \mathbf{c}' = \mathbf{c}''_2$), but $\mathbf{inj}_{\mathbf{Int}} \neq \mathbf{inj}_X$. On the other hand, evidence $\{(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{refl}_{\mathbf{Int}})\}$ is more precise than $\{(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{inj}_X)\}$, because we can choose $\mathbf{c} = \mathbf{c}_t = \mathbf{refl}_{\mathbf{Int}}$, and $\mathbf{c}' = \mathbf{inj}_X$, such that the diagram commutes: $\mathbf{refl}_{\mathbf{Int}}; \mathbf{inj}_X = \mathbf{inj}_X$ ($\mathbf{c}_t; \mathbf{c}'_2 = \mathbf{c}''_2$), and $\mathbf{refl}_{\mathbf{Int}}; \mathbf{inj}_X = \mathbf{inj}_X$ ($\mathbf{c}'_1; \mathbf{c}' = \mathbf{c}''_2$).

Note that the evidence precision judgment $[\mathbf{c}]\varepsilon_1 \sqsubseteq \varepsilon_2[\mathbf{c}']$ explicitly tracks proof terms \mathbf{c} and \mathbf{c}' (we will refer to them as *boundary proofs*). The reason is that monotonicity of consistent transitivity only holds when adjacent boundary proofs match up.

Lemma 6.12 (\checkmark Monotonicity of Consistent Transitivity) *If $[\mathbf{c}]\varepsilon_1 \sqsubseteq \varepsilon_2[\mathbf{c}']$, $[\mathbf{c}']\varepsilon'_1 \sqsubseteq \varepsilon'_2[\mathbf{c}'']$ and $(\varepsilon_1 \mathbin{\text{\textcircled{;}}} \varepsilon'_1)$ is defined, then $[\mathbf{c}](\varepsilon_1 \mathbin{\text{\textcircled{;}}} \varepsilon'_1) \sqsubseteq (\varepsilon_2 \mathbin{\text{\textcircled{;}}} \varepsilon'_2)[\mathbf{c}'']$.*

Let us consider the following example to understand why the “middle” boundary proof terms must match. We have that:

$$[\mathbf{refl}_{\mathbf{Int}}]\{(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{refl}_{\mathbf{Int}})\} \sqsubseteq \{(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{inj}_X)\}[\mathbf{inj}_X]$$

and

$$[\mathbf{inj}_{\mathbf{Int}}]\{(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{refl}_{\mathbf{Int}})\} \sqsubseteq \{(\mathbf{Int}, \mathbf{inj}_{\mathbf{Int}}, \mathbf{refl}_{\mathbf{Int}})\}[\mathbf{refl}_{\mathbf{Int}}]$$

The precision proofs do not match ($\mathbf{inj}_X \neq \mathbf{inj}_{\mathbf{Int}}$), and even though $\{(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{refl}_{\mathbf{Int}})\} \mathbin{\text{\textcircled{;}}} \{(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{refl}_{\mathbf{Int}})\}$ is defined, $\{(\mathbf{Int}, \mathbf{refl}_{\mathbf{Int}}, \mathbf{inj}_X)\} \mathbin{\text{\textcircled{;}}} \{(\mathbf{Int}, \mathbf{inj}_{\mathbf{Int}}, \mathbf{refl}_{\mathbf{Int}})\}$ is not.

Similar to consistent transitivity, type substitution over evidence is also monotonous concerning evidence precision (two evidences related by precision remain related after type substitution).

$\Omega \vdash c : s \sqsubseteq s$ Term precision

$$\begin{array}{c}
\frac{}{\sqsubseteq_b \Omega \vdash \text{refl}_B : b \sqsubseteq b} \qquad \frac{}{\sqsubseteq_x \Omega(x) = c : G_1 \sqsubseteq G_2 \quad \Omega \vdash c : x \sqsubseteq x} \\
\frac{}{\sqsubseteq_\lambda \Omega, x \mapsto c : G'_1 \sqsubseteq G'_2 \vdash c' : t_1 \sqsubseteq t_2 \quad \Omega \vdash c \rightarrow c' : \lambda x : G'_1.t_1 \sqsubseteq \lambda x : G'_2.t_2} \qquad \frac{}{\sqsubseteq_\Lambda \Omega \vdash c : t_1 \sqsubseteq t_2 \quad \Omega \vdash \forall X.c : \Lambda X.t_1 \sqsubseteq \Lambda X.t_2} \\
\frac{}{\sqsubseteq_{\times} \Omega \vdash c : s_1 \sqsubseteq s_2 \quad \Omega \vdash c' : s'_1 \sqsubseteq s'_2 \quad \Omega \vdash c \times c' : \langle s_1, s'_1 \rangle \sqsubseteq \langle s_2, s'_2 \rangle} \qquad \frac{}{\sqsubseteq_{\text{pairi}} \Omega \vdash c_1 \times c_2 : t_1 \sqsubseteq t_2 \quad \Omega \vdash c_i : \pi_i(t_1) \sqsubseteq \pi_i(t_2)} \\
\frac{}{\sqsubseteq_{\text{app}} \Omega \vdash c' \rightarrow c : t_1 \sqsubseteq t_2 \quad \Omega \vdash c' : t'_1 \sqsubseteq t'_2 \quad \Omega \vdash c : t_1 t'_1 \sqsubseteq t_2 t'_2} \qquad \frac{}{\sqsubseteq_{\text{appG}} \Omega \vdash \forall X.c : t_1 \sqsubseteq t_2 \quad \Omega \vdash c[F/X] : t_1[F] \sqsubseteq t_2[F]} \\
\frac{}{\sqsubseteq_{\text{asc}} \Omega \vdash c' : s_1 \sqsubseteq s_2 \quad c : G_1 \sqsubseteq G_2 \quad [c']_{\varepsilon_1} \sqsubseteq \varepsilon_2[c] \quad \Omega \vdash c : \varepsilon_1 s_1 :: G_1 \sqsubseteq \varepsilon_2 s_2 :: G_2}
\end{array}$$

Figure 6.7: $F_\varepsilon^?$: Term Precision.

Lemma 6.13 (Monotonicity of Type Substitution) *If $[\forall X.c']_{\varepsilon_1} \sqsubseteq \varepsilon_2[\forall X.c]$, then $[c'](schm(\varepsilon_1)) \sqsubseteq (schm(\varepsilon_2))[c]$ and $[c'[F/X]](schm(\varepsilon_1)[F/X]) \sqsubseteq (schm(\varepsilon_2)[F/X])[c[F/X]]$.*

6.5.2 Term precision

Term precision is the natural lifting of type and evidence precision to terms, and is presented in Figure 6.7. Judgment $\Omega \vdash c : s_1 \sqsubseteq s_2$ denotes that term s_1 is more precise than s_2 justified by proof term c , under precision relation environment Ω . Boundary proof terms c are propagated for types, contexts, evidence, and subterms, justifying that the type of the less precise term is less precise than the type of the more precise term. Ω binds a term variable x to a type precision judgment $c : G_1 \sqsubseteq G_2$. Rule (\sqsubseteq_x) establishes that a term variable is related to itself along boundary proof term c if $x : (c : G_1 \sqsubseteq G_2) \in \Omega$, and Rule (\sqsubseteq_λ) extends Ω with the judgment that justifies that the argument types are in precision. Analogous to MCT, rule (\sqsubseteq_{app}) requires that the domain proof term of the function matches with the proof term of the arguments. Rule (\sqsubseteq_{asc}) establishes that two ascriptions are related if the sub-terms, s_1 and s_2 , are in precision with proof term c' , the ascribed types, G_1 and G_2 , are in precision with the proof c , and evidences are in precision with the boundary proof terms c' and c .

Gradual guarantees. Armed with the definition of term precision, we now establish the graduality of $F_\varepsilon^?$ with the gradual guarantees (Siek et al., 2015a).

Theorem 6.14 *Suppose $\vdash t_1 : G_1$ and $\vdash c : t_1 \sqsubseteq t_2$. Then,*

- $\vdash t_2 : G_2$ and $c : G_1 \sqsubseteq G_2$.
- $t_1 \xrightarrow{*} v_1$ implies $t_2 \xrightarrow{*} v_2$ and $\vdash c : v_1 \sqsubseteq v_2$.
- t_1 diverges implies t_2 diverges.

The only peculiarity of this result compared to others in the literature is that the type and term precision judgments are proof-relevant. The static part of graduality (the static gradual guarantee) ensures that if t_1 with type G_1 is more precise than t_2 , justified by proof c , then t_2 has a less precise type G_2 justified by c . The dynamic part of graduality (the dynamic gradual guarantee) establishes that if the more precise term reduces to a value, then the less precise term also does, resulting in values in precision with the same type proof term c . The key lemmas to prove graduality, are MCT (Lemma 6.12), and monotonicity of type substitution (evidence precision is monotonous with respect to type substitution) (Lemma 6.13).

6.6 Related Work

We have already discussed in detail related work to gradual parametricity approaches. Thus, this chapter discusses related work to some technical features of $F_\varepsilon^?$, such as evidence representation and proof terms for type precision.

Sealing for Parametricity. Dynamic sealing, originally proposed by Morris (1973) to dynamically enforce type abstraction, has been widely used to guarantee parametricity in gradual languages. The notion of dynamic sealing combined with global runtime type name generation has driven the dynamic semantics of polymorphic gradual languages such as λB (Ahmed et al., 2017), CSA (Xie et al., 2018), GSF (Toro et al., 2019) and PolyG $^\nu$ (New et al., 2020). Type names are dynamically generated in each type application and are kept in a global store, making the dynamic semantics and the definitions and proof of parametricity less standard and more complex. $F_\varepsilon^?$ avoids using type names generation and, therefore, a global store thanks to the fact that the unknown type is decorated by an environment and the sealing/unsealing mechanism is generated statically. It is worth noting that, unlike other developed gradual polymorphic languages, PolyG $^\nu$ also includes explicit seal and unseal terms in its syntax. In this sense, we can say that $F_\varepsilon^?$ also includes in its syntax explicit forms of sealing and unsealing, since for a program with an imprecise type to behave in a parametric way, it is necessary to introduce the evidence of sealing and unsealing statically. The key novelty of $F_\varepsilon^?$ is to support evidence with *multiple* sealing justifications, which makes it possible to avoid to eagerly choose a sealing strategy when interacting with the unknown type.

Proof Terms for Type Precision. PolyG $^\nu$, inspired by previous work (New and Ahmed, 2018), adds a proof term to the type precision relation as a technical intermediate representation for the translation from PolyG $^\nu$ to PolyC $^\nu$, a cast calculus that gives meaning to PolyG $^\nu$ programs. It is important to note that proof terms in the type precision relation are canonical, *i.e.* there is at most one proof term that proves any given type precision judgment. Likewise, $F_\varepsilon^?$ language indexes the type precision relation with proof terms, but contrary to PolyG $^\nu$, proof terms are relevant, *i.e.* there can be multiple proof terms for the same precision judgment. Also, as a difference with PolyG $^\nu$, term precision in $F_\varepsilon^?$ is indexed by a relevant proof term.

Evidence Representation. Evidence has been used in different scenarios, varying its representation according to the semantic properties to be preserved in the gradual language. For instance, Lehmann and Tanter (2017) develop a gradual language with refinement types, allowing smooth evolution and interoperability between simple types and logically refined types. In this case, the evidence for consistent subtyping is represented by a triple, where the first component accounts for the logical environment, and the second and third are types. Toro et al. (2018) develop a gradual language with security types and references, indexing types with gradual security labels. Driven by noninterference, types in evidence are indexed with *intervals* of security labels, representing

(bounded) ranges of possible static types. Likewise, $F_\varepsilon^?$ represents evidence in a novel way. First, it enriches evidence with proof terms relevant that we call spans and then generalizes the evidence to a set of spans, building evidence with the expressiveness to ensure both graduality and parametricity. This theory may be applicable in other complex settings as well. In addition, it would be interesting to explore if there is a way to systematically derive proof-relevant consistency with AGT.

Performance. Gradual parametricity is a very challenging topic at the theoretical level, with all current efforts trying to figure out how to achieve a good design backed by a strong metatheory. This work likewise focuses on the theory of gradual parametricity, contributing a novel approach and technique. We leave the study of the performance and efficiency of a practical implementation as an open question to be addressed. Nevertheless, it is worth mentioning that the proposed language design satisfies a relevant criterion for space efficiency (Herman et al., 2010), namely associativity of evidence composition, which is known to allow for space efficiency in evidence-based semantics (Bañados Schwerter et al., 2021; Toro and Tanter, 2020). Whether the algorithmic definition of consistent transitivity can be efficiently implemented depends on whether evidence can be represented in memory in a form that uses space efficiently and allows an efficient implementation of evidence composition. All these are open research questions.

6.7 Conclusion

Previous work on gradual parametricity has had to compromise on important design goals like graduality (Toro et al., 2019) or type-driven sealing (New et al., 2020). Rather than accepting these compromises, this work attempts to revisit accepted wisdom like the use of globally scoped sealing and contribute new ideas like plausible sealing and the set-of-spans representation of evidence for proof-relevant precision. Although the results presented here still have some restrictions, they open a new path towards the goal of reconciling parametricity, graduality and type-driven sealing. Additionally, some of our novel techniques are potentially reusable in other settings. Evidence represented as a set of spans could be used to verify hyperproperties (sets of properties) (Clarkson and Schneider, 2008) in programming languages. For instance, in security-typed languages, types statically enforce the noninterference hyperproperty. Toro et al. (2018) established noninterference for a gradual security-typed language with references but sacrificing graduality. It would be interesting to investigate if a gradual language with those features can benefit from our techniques in order to recover graduality. Also, it could be explored on the gradualization of a probabilistic language, where it is needed to track different path execution with different types.

In the next chapter, we present the source language $F^?$ and its elaboration to $F_\varepsilon^?$, illustrating the practicality of $F_\varepsilon^?$ as a target language for a type-driving source language. We establish graduality for $F^?$, and explain the source-level parametric reasoning that $F^?$ offers. Finally, we discuss lifting the technical restrictions of this work for both $F_\varepsilon^?$ and $F^?$.

Chapter 7

The Gradual Source Language $F^?$

Having formalized the key technical innovation of this work, plausible sealing, and established both graduality and parametricity for the intermediate language $F_\epsilon^?$, we now turn to the source language $F^?$ and its properties. We formalize the source language $F^?$ and its elaboration to $F_\epsilon^?$. The novel translation to $F_\epsilon^?$ plays a crucial role since it is in charge of statically generating the maybe-sealing evidence for type applications. We study the gradual guarantees for $F^?$ and the resulting source-level parametric reasoning. It is worth pointing out that reasoning about graduality in $F^?$ requires users to verify that the types of polymorphic functions being applied have the same shape; for instance, graduality holds between functions of types $\forall X.X \rightarrow X$ and $\forall X.? \rightarrow ?$, but not between $\forall X.X \rightarrow X$ and $\forall X.?$. Except for this technical restriction, type and term precision is standard and graduality in $F^?$ allows programmers to reason in much the same way as they would with the natural notion of term precision.

Overview. This chapter presents the static semantics of $F^?$ (Section 7.1) and its translation to $F_\epsilon^?$ (Section 7.2). We establish graduality for $F^?$ (Section 7.3), and explain the source-level parametric reasoning that $F^?$ offers (Section 7.4). We discuss the lifting of the technical restrictions of this work in Section 7.5 and Section 7.6 concludes.

7.1 $F^?$: Statics

The static semantics of $F^?$ is derived systematically by applying AGT to System F_1 (Figure 7.1), which is a variation of System F where type instantiations are restricted to instantiation types F (*i.e.* base types and type variables).

In order to apply AGT to obtain the static semantics (*i.e.* lifting functions and predicates), we use explicit type equalities and partial type functions in the typing rules of System F_1 (as in Chapter 3). These partial functions also allow capturing elimination forms in a single rule that accounts for both precise and imprecise type information (Garcia et al., 2016).

The syntactic category of source gradual types G admits the unknown type in any position (like GSF), namely:

$$G ::= B \mid X \mid G \rightarrow G \mid G \times G \mid \forall X.G \mid ?_\delta \quad (\text{source gradual types})$$

$$\begin{aligned}
F & ::= B \mid X \\
T & ::= B \mid X \mid T \rightarrow T \mid T \times T \mid \forall X.T \\
t & ::= b \mid \lambda x : T.t \mid \Lambda X.t \mid x \mid \langle t, t \rangle \mid t t \mid \pi_i(t) \mid t [T] \mid t :: T
\end{aligned}$$

$\Delta; \Gamma \vdash t : T$ **Term typing**

$$\begin{array}{c}
(T_x) \frac{x : T \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x : T} \quad (T_b) \frac{\theta(b) = B \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash b : B} \quad (T_\lambda) \frac{\Delta; \Gamma, x : T_1 \vdash t : T_2}{\Delta; \Gamma \vdash \lambda x : T_1. t : T_1 \rightarrow T_2} \\
(T_{\text{pair}}) \frac{\Delta; \Gamma \vdash t_1 : T_1 \quad \Delta; \Gamma \vdash t_2 : T_2}{\Delta; \Gamma \vdash \langle t_1, t_2 \rangle : T_1 \times T_2} \quad (T_\Lambda) \frac{\Delta, X; \Gamma \vdash t : T \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \Lambda X. t : \forall X. T} \\
(T_{\text{asc}}) \frac{\Delta; \Gamma \vdash t : T' \quad T' = T}{\Delta; \Gamma \vdash t :: T : T} \quad (T_{\text{app}}) \frac{\Delta; \Gamma \vdash t_1 : T_1 \quad \Delta; \Gamma \vdash t_2 : T_2 \quad \text{dom}(T_1) = T_2}{\Delta; \Gamma \vdash t_1 t_2 : \text{cod}(T_1)} \\
(T_{\text{pair}i}) \frac{\Delta; \Gamma \vdash t : T}{\Delta; \Gamma \vdash \pi_i(T) : \text{proj}_i(T)} \quad (T_{\text{app}G}) \frac{\Delta; \Gamma \vdash t : T \quad \Delta \vdash F}{\Delta; \Gamma \vdash t [F] : \text{inst}(T, F)}
\end{array}$$

$t \longrightarrow t$ **Notion of reduction**

$$\begin{aligned}
v & :: T \longrightarrow v \\
(\lambda x : T. t) v & \longrightarrow t[v/x] \\
\pi_i(\langle v_1, v_2 \rangle) & \longrightarrow v_i \\
(\Lambda X. t) [F] & \longrightarrow t[F/X]
\end{aligned}$$

$t \mapsto t$ or **error** **Evaluation frames and reduction**

$$\begin{array}{c}
f ::= \square :: T \mid \square t \mid v \square \mid \langle \square, t \rangle \mid \langle v, \square \rangle \mid \pi_i(\square) \mid \square [F] \\
(R_{\rightarrow}) \frac{t \longrightarrow t'}{t \mapsto t'} \quad (R_f) \frac{t \mapsto t'}{f[t] \mapsto f[t']} \\
(R_{\text{err}}) \frac{t \longrightarrow \mathbf{error}}{t \mapsto \mathbf{error}} \quad (R_{f\text{err}}) \frac{t \mapsto \mathbf{error}}{f[t] \mapsto \mathbf{error}}
\end{array}$$

Figure 7.1: System F_1 : Static Polymorphic Language Restricted to Instantiation Types F

Source gradual types are syntactically contained in the gradual types of $F_\varepsilon^?$, and for simplicity throughout this section, we write \mathbf{G} as the $F_\varepsilon^?$ counterpart of \mathbf{G} . Source gradual types restricts the scope of unknown types to not-instantiated variables only. To represent this, we index unknown types with the δ meta-variable (included in δ); and as every type variable in $?_\delta$ is not instantiated (i.e. of the form $X : X$), for simplicity we just write δ as a set of type variables. One final note is that $?$ (without a scope δ , as used in previous sections) is syntactic sugar for $?_\Delta$, where Δ is the set of all variables in scope at that point. A straightforward and simple translation can insert these

$$\begin{array}{ll}
\gamma(\mathbf{B}) = \{B\} & \alpha(\{B\}) = \mathbf{B} \\
\gamma(\mathbf{X}) = \{X\} & \alpha(\{X\}) = \mathbf{X} \\
\gamma(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}_2) & \alpha(\{\overline{T_1 \rightarrow T_2}\}) = \alpha(\{\overline{T_1}\}) \rightarrow \alpha(\{\overline{T_2}\}) \\
\gamma(\mathbf{G}_1 \times \mathbf{G}_2) = \gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}_2) & \alpha(\{\overline{T_1 \times T_2}\}) = \alpha(\{\overline{T_1}\}) \times \alpha(\{\overline{T_2}\}) \\
\gamma(\forall X. \mathbf{G}_1) = \forall X. \gamma(\mathbf{G}_1) & \alpha(\{\overline{\forall X. T_1}\}) = \forall X. \alpha(\{\overline{T_1}\}) \\
\gamma(?_\delta) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\} & \alpha(\{\overline{T}\}) = ?_\delta \quad \text{where } \text{ftv}(\delta) = \text{ftv}(\{\overline{T}\})
\end{array}$$

Figure 7.2: Type concretization (γ) and abstraction (α)

annotations before typing.

We give meaning to gradual types \mathbf{G} through the concretization function $\gamma(\cdot)$, presented in Figure 7.2. The meaning of the unknown type $?_\delta$ is the set of all well-formed static types with respect to δ (i.e. $\gamma(?_\delta) = \{T \mid \delta \vdash T\}$). The concretization function helps us define precision and consistency.

Definition 7.1 (Type Precision) $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$ if and only if $\gamma(\mathbf{G}_1) \subseteq \gamma(\mathbf{G}_2)$.

Definition 7.2 (Type Consistency) $\mathbf{G}_1 \sim \mathbf{G}_2$ if and only if there exists T_1 and T_2 such that $T_1 = T_2$, $T_1 \in \gamma(\mathbf{G}_1)$ and $T_2 \in \gamma(\mathbf{G}_2)$.

$\mathbf{G} \sim \mathbf{G}$ Type Consistency

$$\begin{array}{c}
\frac{}{B \sim B} \quad \frac{}{X \sim X} \quad \frac{G_1 \sim G'_1 \quad G_2 \sim G'_2}{G_1 \times G_2 \sim G'_1 \times G'_2} \quad \frac{G'_1 \sim G_1 \quad G_2 \sim G'_2}{G_1 \rightarrow G_2 \sim G'_1 \rightarrow G'_2} \\
\frac{G \sim G'}{\forall X. G \sim \forall X. G'} \quad \frac{}{B \sim ?_\delta} \quad \frac{}{?_\delta \sim B} \quad \frac{X \in \delta}{X \sim ?_\delta} \quad \frac{X \in \delta}{?_\delta \sim X} \\
\frac{G_1 \sim ?_\delta \quad G_2 \sim ?_\delta}{G_1 \rightarrow G_2 \sim ?_\delta} \quad \frac{G_1 \sim ?_\delta \quad G_2 \sim ?_\delta}{G_1 \times G_2 \sim ?_\delta} \quad \frac{G \sim ?_{\delta, X}}{\forall X. G \sim ?_\delta} \quad \frac{?_\delta \sim G_1 \quad ?_\delta \sim G_2}{?_\delta \sim G_1 \rightarrow G_2} \\
\frac{?_\delta \sim G_1 \quad ?_\delta \sim G_2}{?_\delta \sim G_1 \times G_2} \quad \frac{?_{\delta, X} \sim G}{?_\delta \sim \forall X. G} \quad \frac{}{?_\delta \sim ?_{\delta'}}
\end{array}$$

$\mathbf{G} \sqsubseteq \mathbf{G}$ Type Precision

$$\begin{array}{c}
\sqsubseteq_B \frac{}{B \sqsubseteq B} \quad \sqsubseteq_X \frac{}{X \sqsubseteq X} \quad \sqsubseteq_{\rightarrow} \frac{G_1 \sqsubseteq G_2 \quad G'_1 \sqsubseteq G'_2}{G_1 \rightarrow G'_1 \sqsubseteq G_2 \rightarrow G'_2} \quad \sqsubseteq_{\times} \frac{G_1 \sqsubseteq G_2 \quad G'_1 \sqsubseteq G'_2}{G_1 \times G'_1 \sqsubseteq G_2 \times G'_2} \\
\sqsubseteq_{\forall} \frac{G_1 \sqsubseteq G_2}{\forall X. G_1 \sqsubseteq \forall X. G_2} \quad \sqsubseteq_{B?} \frac{}{B \sqsubseteq ?_\delta} \quad \sqsubseteq_{X?} \frac{X \in \delta}{X \sqsubseteq ?_\delta} \quad \sqsubseteq_{\rightarrow?} \frac{G_1 \rightarrow G_2 \sqsubseteq ?_\delta \rightarrow ?_\delta}{G_1 \rightarrow G_2 \sqsubseteq ?_\delta} \\
\sqsubseteq_{\times?} \frac{G_1 \times G_2 \sqsubseteq ?_\delta \times ?_\delta}{G_1 \times G_2 \sqsubseteq ?_\delta} \quad \sqsubseteq_{\forall?} \frac{\forall X. G \sqsubseteq \forall X. ?_{\delta, X}}{\forall X. G \sqsubseteq ?_\delta} \quad \sqsubseteq_{?} \frac{\delta \sqsubseteq \delta'}{?_\delta \sqsubseteq ?_{\delta'}}
\end{array}$$

Figure 7.3: $F^?$: Type Precision and Consistency Inductively.

Precision and consistency resemble their $F_\varepsilon^?$ counterpart and can also be inductively defined (Figure 7.3). For instance, $X \sim ?_X$, but $X \not\sim ?_Y$ (for $X \neq Y$). However, precision in $F^?$ is no longer proof relevant: $F^?$ contains only unknown types $?_\delta$ with uninstantiated type variables, so that the precision relation from $F_\varepsilon^?$ (which had the structure of a category) reduces to a proof-irrelevant order relation in $F^?$. The inductive definitions of type precision and consistency are equivalent to the derived by AGT.

Proposition 7.3 (Type Precision, inductively) *The inductive definition of type precision given in Figure 7.3 is equivalent to Definition B.94.*

Proposition 7.4 (Type Consistency, inductively) *The inductive definition of type consistency given in Figure 7.3 is equivalent to Definition 7.2.*

Figure 7.4 presents the term typing rules for $F^?$, which are obtained by replacing type predicates and functions with their corresponding liftings. The lifting of function and predicates are straightforward and uses the corresponding abstraction function $\alpha(\cdot)$ of $\gamma(\cdot)$ (Figure 7.2), forming a Galois connection.

Lemma 7.5 (α is Sound) *If A is not empty, then $A \subseteq \gamma(\alpha(A))$.*

Lemma 7.6 (α is Optimal) *If A is not empty and $A \subseteq \gamma(G)$ then $\alpha(A) \sqsubseteq G$.*

For example, rule (Gasc) uses type consistency instead of type equality, and rule (GappG) uses the lifting of the function *inst*, defined for polymorphic types and the unknown type (*i.e.* $inst^\#(\forall X.G, G') = G[G'/X] \setminus X$, $inst^\#(?_\delta, G') = ?_\delta$ and undefined for other cases). Note that $inst^\#$ uses the scope removal function $G \setminus X$, which removes X from the scopes of unknown types in G : $?_{\delta_1, X, \delta_2} \setminus X = ?_{\delta_1, \delta_2}$. For instance, $(X \rightarrow ?_{X, Y})[Int/X] \setminus X = Int \rightarrow ?_Y$.

7.2 $F^?$: Elaboration to $F_\varepsilon^?$

The dynamic semantics of a $F^?$ program is given by a type-directed translation to $F_\varepsilon^?$. Figure B.7 describes the type-preserving elaboration of $F^?$ terms to $F_\varepsilon^?$. The elaboration rules are type-directed, following the type rules for $F^?$. The rules are mostly standard save for the elaboration rule for type application. Judgment $\Delta; \Gamma \vdash t : G \rightsquigarrow t'$ expresses that term t is elaborated to t' , under type variable environment Δ , and type environment Γ .

The elaboration rules use the function $initEv(G_1, G_2)$, which stands for the *initial evidence* between G_1 and G_2 . It computes the least precise evidence that justifies consistency between the types, and is defined as follows:

Definition 7.7 (Initial Evidence) *If $G_1 \sim G_2$ then $G = G_1 \sqcap G_2$ and $initEv(G_1, G_2) = \{(G, initPT(G, G_1), initPT(G, G_2))\}$.*

This evidence consists of a single span, where the first component is the *meet* (greatest lower bound with respect to precision) $G_1 \sqcap G_2$ between G_1 and G_2 , and the second and third components are the *initial proof terms* between the meet and G_1 and G_2 , respectively (Figure 7.6). The meet $G_1 \sqcap G_2$ is a partial function and corresponds formally to $\alpha(\gamma(G_1) \cap \gamma(G_2))$. Note that from AGT,

$\delta ::= \delta, X : X \mid \cdot$
 $F ::= B \mid X$ (simple types)
 $G ::= B \mid X \mid G \rightarrow G \mid G \times G \mid \forall X. G \mid ?_\delta$ (source gradual types)
 $t ::= b \mid \lambda x : G. t \mid \Lambda X. t \mid x \mid \langle t, t \rangle \mid t t \mid \pi_i(t) \mid t [F] \mid t :: G$ (source gradual terms)

$\boxed{\Delta; \Gamma \vdash t : G}$ **Term typing**

$$\begin{array}{c}
G_x \frac{x : G \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x : G} \quad G_b \frac{\theta(b) = B \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash b : B} \quad G_\lambda \frac{\Delta; \Gamma, x : G_1 \vdash t : G_2}{\Delta; \Gamma \vdash \lambda x : G_1. t : G_1 \rightarrow G_2} \\
G_\times \frac{\Delta; \Gamma \vdash t_1 : G_1 \quad \Delta; \Gamma \vdash t_2 : G_2}{\Delta; \Gamma \vdash \langle t_1, t_2 \rangle : G_1 \times G_2} \quad G_\Lambda \frac{\Delta, X; \Gamma \vdash t : G \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \Lambda X. t : \forall X. G} \\
G_{\text{asc}} \frac{\Delta; \Gamma \vdash t : G' \quad \Delta \vdash G \quad G' \sim G}{\Delta; \Gamma \vdash t :: G : G} \\
G_{\text{app}} \frac{\Delta; \Gamma \vdash t_1 : G_1 \quad \Delta; \Gamma \vdash t_2 : G_2 \quad \text{dom}^\sharp(G_1) \sim G_2}{\Delta; \Gamma \vdash t_1 t_2 : \text{cod}^\sharp(G_1)} \quad G_{\text{pair}_i} \frac{\Delta; \Gamma \vdash t : G}{\Delta; \Gamma \vdash \pi_i(t) : \text{proj}_i^\sharp(G)} \\
G_{\text{app}G} \frac{\Delta; \Gamma \vdash t : G \quad \Delta \vdash F}{\Delta; \Gamma \vdash t [F] : \text{inst}^\sharp(G, F)}
\end{array}$$

Figure 7.4: $F^?$: Syntax and Static Semantics.

$G \sim G'$ holds if $\gamma(G_1) \cap \gamma(G_2)$ is not empty, then if $G \sim G'$ then $G \sqcap G'$ will always be defined.

Definition 7.8 (Meet) $G_1 \sqcap G_2 = \alpha(\gamma(G_1) \cap \gamma(G_2))$.

The initial proof term between two types in precision is computed using the $\text{initPT}(G, G')$ function such that $\text{initPT}(G, G') : G \sqsubseteq G'$. It is important to note that the initial proof term between two types is unique since the type variables within the unknown type scope are not instantiated. Its definition is unsurprising and can be derived from the type precision judgment from Figure 6.2. The initial proof term between two base types B or two type variable X are the reflexive proof terms B and X , respectively. The initial proof term between a base type and the unknown type is an injection from that base type inj_B . The cases for type variables are defined analogously. The initial proof term between a function, a pair, or a polymorphic function and the unknown type, is computed by composing the initial proof term of the precise type and a ground type that matches its constructor, and an injection to unknown. The initial proof term between two unknown types is just $\text{inj}_?$, and the rest of the cases are defined inductively. For example, the initial proof term that justifies judgment $\text{Int} \sqsubseteq ?_X$ is inj_{Int} , for $X \sqsubseteq ?_X$ is inj_X , for $?_X \sqsubseteq ?_{X,Y}$ is $\text{inj}_?$, and for $\text{Int} \rightarrow ?_X \sqsubseteq ?_{X,Y}$ is $\text{inj}_{\rightarrow}(\text{inj}_{\text{Int}} \rightarrow \text{inj}_?)$.

As $F_e^?$ requires all values to be ascribed, the elaboration rules ascribe base values, functions and type abstractions to their own type using the *reflexive evidence* operator $\text{reflEv}(G) \triangleq \text{initEv}(G, G)$ (like the translation from GSF to GSF ε). For instance, term 42 is elaborated to $\text{reflEv}(\text{Int}) \text{42} :: \text{Int}$, where $\text{reflEv}(\text{Int}) = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{refl}_{\text{Int}})\}$. The elaboration process also inserts ascriptions to equate types in elimination forms. In particular, the translation of a term application (Eapp) ascribes the function term t_1 to a function type that matches (Cimini and Siek, 2016) with its own

$\Delta; \Gamma \vdash t : G \rightsquigarrow t$ **Elaboration rules**

$$\begin{array}{c}
\text{Eb} \frac{\theta(b) = B \quad \Delta \vdash \Gamma \quad \varepsilon = \text{reflEv}(B)}{\Delta; \Gamma \vdash b : B \rightsquigarrow \varepsilon b :: B} \qquad \text{Ex} \frac{x : G \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x : G \rightsquigarrow x} \\
\\
\text{E}_\times \frac{\Delta; \Gamma \vdash t_1 : G_1 \rightsquigarrow t'_1 \quad \Delta; \Gamma \vdash t_2 : G_2 \rightsquigarrow t'_2}{\Delta; \Gamma \vdash \langle t_1, t_2 \rangle : G_1 \times G_2 \rightsquigarrow \langle t'_1, t'_2 \rangle} \\
\\
\text{E}_\lambda \frac{\Delta; \Gamma, x : G_1 \vdash t : G_2 \rightsquigarrow t' \quad \varepsilon = \text{reflEv}(G_1 \rightarrow G_2)}{\Delta; \Gamma \vdash \lambda x : G_1. t : G_1 \rightarrow G_2 \rightsquigarrow \varepsilon (\lambda x : G_1. t') :: G_1 \rightarrow G_2} \\
\\
\text{E}_\Lambda \frac{\Delta, X; \Gamma \vdash t : G \rightsquigarrow t' \quad \Delta \vdash \Gamma \quad \varepsilon = \text{reflEv}(\forall X. G)}{\Delta; \Gamma \vdash \Lambda X. t : \forall X. G \rightsquigarrow \varepsilon (\Lambda X. t') :: \forall X. G} \\
\\
\text{Easc} \frac{\Delta; \Gamma \vdash t : G' \rightsquigarrow t' \quad \Delta \vdash G \quad \varepsilon = \text{initEv}(G', G)}{\Delta; \Gamma \vdash t : G : G \rightsquigarrow \varepsilon t' :: G} \\
\\
\text{Eapp} \frac{\Delta; \Gamma \vdash t_1 : G_1 \rightsquigarrow t'_1 \quad \Delta; \Gamma \vdash t_2 : G_2 \rightsquigarrow t'_2 \quad G_1 \rightarrow G_{11} \rightarrow G_{12} \quad \varepsilon_1 = \text{initEv}(G_1, G_{11} \rightarrow G_{12}) \quad \varepsilon_2 = \text{initEv}(G_2, G_{11})}{\Delta; \Gamma \vdash t_1 t_2 : G_{12} \rightsquigarrow (\varepsilon_1 t'_1 :: G_{11} \rightarrow G_{12}) (\varepsilon_2 t'_2 :: G_{11})} \\
\\
\text{EappG} \frac{\Delta; \Gamma \vdash t : G \rightsquigarrow t' \quad \Delta \vdash F \quad G \rightarrow \forall X. G' \quad \varepsilon_1 = \text{initEv}(G, \forall X. G') \quad \varepsilon_2 = \text{instEv}(G', X, F)}{\Delta; \Gamma \vdash t [F] : G'[F/X] \setminus X \rightsquigarrow \varepsilon_2 ((\varepsilon_1 t' :: \forall X. G') [F']) :: G'[F'/X] \setminus X} \\
\\
\text{Epairi} \frac{\Delta; \Gamma \vdash t : G \rightsquigarrow t' \quad G \rightarrow G_1 \times G_2 \quad \varepsilon = \text{initEv}(G, G_1 \times G_2)}{\Delta; \Gamma \vdash \pi_i(t) : G_i \rightsquigarrow \pi_i(\varepsilon t' :: G_1 \times G_2)}
\end{array}$$

$G \rightarrow G$ **Type matching**

$$\begin{array}{c}
G_1 \rightarrow G_2 \rightarrow G_1 \rightarrow G_2 \qquad \forall X. G \rightarrow \forall X. G \qquad G_1 \times G_2 \rightarrow G_1 \times G_2 \qquad ?_\delta \rightarrow ?_\delta \rightarrow ?_\delta \qquad ?_\delta \rightarrow \forall X. ?_{\delta, X} \\
?_\delta \rightarrow ?_\delta \times ?_\delta
\end{array}$$

Figure 7.5: Translation from $F^?$ to $F_\varepsilon^?$

type $(G_1 \rightarrow G_{11} \rightarrow G_{12})$: its own type if t_1 has a function type; otherwise $?_\delta \rightarrow ?_\delta$ (if the type is $?_\delta$). Also, the argument term t_2 is ascribed to the argument type of the ascribed function. Rule (Easc) translates ascriptions by first translating the sub-term t , and then computing the initial evidence between the type of the sub-term (G') and the ascription type (G).

Rule (EappG) elaborates type applications, and is responsible of inserting “maybe-seal” evidence. The elaborated type application is ascribed to the instantiated scheme type $G'[F/X] \setminus X$ (removing X from the environments of unknown types), using a special evidence that justifies that $G'[F/X]$ is consistent with $G'[F/X] \setminus X$. This evidence is computed using the *instantiation evidence* function instEv defined in Figure 7.7. Function $\text{instEv}(G, X, F) : G[F/X] \sim G[F/X] \setminus X$ is defined (inductively) almost as the reflexive evidence operator, save for the case when type G is $?_\delta$ and X is in scope δ . Then, instEv generates an evidence consisting of two spans: one span that “seals to X ” and another one that does not. More in detail, the first span $(F, \text{inj}_X,$

$$\begin{aligned}
\text{initPT}(\mathbf{B}, \mathbf{B}) &= \mathbf{B} \\
\text{initPT}(\mathbf{X}, \mathbf{X}) &= \mathbf{X} \\
\text{initPT}(\mathbf{G}_1 \rightarrow \mathbf{G}_2, \mathbf{G}'_1 \rightarrow \mathbf{G}'_2) &= \text{initPT}(\mathbf{G}_1, \mathbf{G}'_1) \rightarrow \text{initPT}(\mathbf{G}_2, \mathbf{G}'_2) \\
\text{initPT}(\mathbf{G}_1 \times \mathbf{G}_2, \mathbf{G}'_1 \times \mathbf{G}'_2) &= \text{initPT}(\mathbf{G}_1, \mathbf{G}'_1) \times \text{initPT}(\mathbf{G}_2, \mathbf{G}'_2) \\
\text{initPT}(\forall \mathbf{X}. \mathbf{G}_1, \forall \mathbf{X}. \mathbf{G}'_1) &= \forall \mathbf{X}. \text{initPT}(\mathbf{G}_1, \mathbf{G}'_1) \\
\text{initPT}(\mathbf{B}, ?_\delta) &= \text{inj}_{\mathbf{B}} \\
\text{initPT}(\mathbf{X}, ?_\delta) &= \text{inj}_{\mathbf{X}} \\
\text{initPT}(\mathbf{G}_1 \rightarrow \mathbf{G}_2, ?_\delta) &= \text{inj}_{\rightarrow}(\text{initPT}(\mathbf{G}_1 \rightarrow \mathbf{G}_2, ?_\delta \rightarrow ?_\delta)) \\
\text{initPT}(\mathbf{G}_1 \times \mathbf{G}_2, ?_\delta) &= \text{inj}_{\times}(\text{initPT}(\mathbf{G}_1 \times \mathbf{G}_2, ?_\delta \times ?_\delta)) \\
\text{initPT}(\forall \mathbf{X}. \mathbf{G}_1, ?_\delta) &= \text{inj}_{\forall}(\text{initPT}(\forall \mathbf{X}. \mathbf{G}_1, \forall \mathbf{X}. ?_\delta, \mathbf{X})) \\
\text{initPT}(?_\delta, ?_{\delta'}) &= \text{inj}_{?}
\end{aligned}$$

Figure 7.6: Initial proof term

$$\begin{aligned}
\text{instEv}(\mathbf{B}, \mathbf{X}, \mathbf{F}) &= \text{reflEv}(\mathbf{B}) \\
\text{instEv}(\mathbf{X}, \mathbf{X}, \mathbf{F}) &= \text{reflEv}(\mathbf{F}) \\
\text{instEv}(?_\delta, \mathbf{X}, \mathbf{F}) &= \text{reflEv}(?_\delta) \text{ if } \mathbf{X} \notin \delta \\
\text{instEv}(\mathbf{Y}, \mathbf{X}, \mathbf{F}) &= \text{reflEv}(\mathbf{Y}) \text{ if } \mathbf{X} \neq \mathbf{Y} \\
\text{instEv}(\mathbf{G}_1 \rightarrow \mathbf{G}_2, \mathbf{X}, \mathbf{F}) &= \text{instEv}(\mathbf{G}_1, \mathbf{X}, \mathbf{F}) \rightarrow \text{instEv}(\mathbf{G}_2, \mathbf{X}, \mathbf{F}) \\
\text{instEv}(\mathbf{G}_1 \times \mathbf{G}_2, \mathbf{X}, \mathbf{F}) &= \text{instEv}(\mathbf{G}_1, \mathbf{X}, \mathbf{F}) \times \text{instEv}(\mathbf{G}_2, \mathbf{X}, \mathbf{F}) \\
\text{instEv}(\forall \mathbf{Y}. \mathbf{G}, \mathbf{X}, \mathbf{F}) &= \forall \mathbf{Y}. \text{instEv}(\mathbf{G}, \mathbf{X}, \mathbf{F}) \\
\text{instEv}(?_{\delta_1; \mathbf{X}; \delta_2}, \mathbf{X}, \mathbf{F}) &= \{(?_{\delta_1; \delta_2}, \text{inj}_{?}, \text{inj}_{?}), (\mathbf{F}, \text{inj}_{\mathbf{X}}, \text{inj}_{\mathbf{F}})\}
\end{aligned}$$

Figure 7.7: Instantiation evidence function.

inj_F) represents that the unknown type should behave polymorphically in \mathbf{X} . On the contrary, the second span $(?_{\delta_1, \delta_2}, \text{inj}_?, \text{inj}_?)$ does not acknowledge the existence of variable \mathbf{X} . For example, we have that $(?_{\mathbf{X} \rightarrow \mathbf{X}})[\text{Int}/\mathbf{X}] = ?_{\mathbf{X}:\text{Int} \rightarrow \text{Int}}$ and $(?_{\mathbf{X} \rightarrow \mathbf{X}})[\text{Int}/\mathbf{X}] \setminus \mathbf{X} = ? \rightarrow \text{Int}$. Therefore $\text{instEv}(?_{\mathbf{X} \rightarrow \mathbf{X}}, \mathbf{X}, \text{Int}) : ?_{\mathbf{X}:\text{Int} \rightarrow \text{Int}} \sim ? \rightarrow \text{Int}$, being defined as follows:

$$\{(\text{Int} \rightarrow \text{Int}, \text{inj}_{\mathbf{X} \rightarrow \text{refl}_{\text{Int}}}, \text{inj}_{\text{Int} \rightarrow \text{refl}_{\text{Int}}}), (? \rightarrow \text{Int}, \text{inj}_? \rightarrow \text{refl}_{\text{Int}}, \text{inj}_? \rightarrow \text{refl}_{\text{Int}})\}$$

Note that this evidence makes it impossible that a sealed value leaks out of a polymorphic function application. The scope of a type variable (label) is limited to the type application in which it appears. For example, consider the type application $\mathbf{t} [\text{Int}]$ in $F^?$, where $\vdash \mathbf{t} : \forall \mathbf{X}. ?_{\mathbf{X}} \rightarrow ?_{\mathbf{X}}$. This term is elaborated to $\text{instEv}(?_{\mathbf{X} \rightarrow ?_{\mathbf{X}}}, \mathbf{X}, \text{Int}) (\mathbf{t} [\text{Int}]) :: ? \rightarrow ?$, where $\mathbf{t} [\text{Int}]$ has type $?_{\mathbf{X}:\text{Int} \rightarrow ?_{\mathbf{X}:\text{Int}}}$; the generated instEv evidence is used to coerce this type to $? \rightarrow ?$. The label \mathbf{X} may appear in unknown types that are used inside $\mathbf{t} [\text{Int}]$, but the instEv evidence casts $\mathbf{t} [\text{Int}]$ to a type not mentioning the label \mathbf{X} , effectively restricting the scope of \mathbf{X} to inside the term $\mathbf{t} [\text{Int}]$. Even when the resulting term computes further and $\mathbf{t} [\text{Int}]$ is reduced and combined with values from the context (for example, in a function application), the instEv evidence protects the scope of \mathbf{X} , preventing it from interfering with possible other occurrences of the same name \mathbf{X} introduced by other type applications. Because of this, there is no need for alpha-renaming.

It is important to clarify one limitation: the instantiation evidence $\text{instEv}(\mathbf{G}, \mathbf{X}, \mathbf{F})$ is not general enough when $\mathbf{G} = ?_{\delta_1, \mathbf{X}, \delta_2}$. With the current definition, $?_{\delta_1, \mathbf{X}, \delta_2}$ intuitively only represents something of type \mathbf{X} or other well-formed static type with respect to δ_1, δ_2 . This means that at runtime, if $?_{\delta_1, \mathbf{X}, \delta_2}$ is used in a consistent judgment with a function such as $\mathbf{X} \rightarrow \mathbf{X}$, the program could fail. For instance, the program $(\lambda \mathbf{X}. \lambda x : \mathbf{X}. x) :: \forall \mathbf{X}. ?_{\mathbf{X}} [\text{Int}] \mathbf{1}$ generates the instantiation evidence $\text{instEv}(?_{\mathbf{X}}, \mathbf{X}, \text{Int}) = \{(? , \text{inj}_?, \text{inj}_?), (\text{Int}, \text{inj}_{\mathbf{X}}, \text{inj}_{\text{Int}})\}$, which will not seal argument $\mathbf{1}$, making this program fail at runtime. To fix the program, and generate appropriate sealing, the type of the ascription had to be changed as follows: $(\lambda \mathbf{X}. \lambda x : \mathbf{X}. x) :: \forall \mathbf{X}. ?_{\mathbf{X}} \rightarrow ?_{\mathbf{X}} [\text{Int}] \mathbf{1}$. The spans generated now include $(\text{Int} \rightarrow \text{Int}, \text{inj}_{\mathbf{X}} \rightarrow \text{inj}_{\mathbf{X}}, \text{inj}_{\text{Int}} \rightarrow \text{inj}_{\text{Int}})$ which makes the program run without errors. However, statically there is no way to know a priori the exact shape of the evidence needed when imprecise information is involved. Consequently, a more general mechanism for the generation of the instantiation evidence is needed (see Section 7.5).

Finally, we prove that the elaboration preserves typing:

Theorem 7.9 (Elaboration Preserves Typing) *If $\Delta; \Gamma \vdash \mathbf{t} : \mathbf{G}$, then $\Delta; \Gamma \vdash \mathbf{t} : \mathbf{G} \rightsquigarrow \mathbf{t}'$ and $\Delta; \Gamma \vdash \mathbf{t}' : \mathbf{G}$.*

Example of elaboration. We now present the elaboration of program 3 from Table 6.1. Program $(\lambda \mathbf{X}. \lambda x : ?_{\mathbf{X}}. x) [\text{Bool}] \mathbf{42}$ is elaborated as follows.

1. As $x : ?_{\mathbf{X}}$, then $x :: \mathbf{X}$ elaborates to $\mathbf{t}_1 = \varepsilon_1 x :: \mathbf{X}$, where $\varepsilon_1 = \{(\mathbf{X}, \text{inj}_{\mathbf{X}}, \text{refl}_{\mathbf{X}})\}$ justifies that $?_{\mathbf{X}} \sim \mathbf{X}$.
2. As all values in $F_\varepsilon^?$ are ascribed, $\lambda x : ?_{\mathbf{X}}. x :: \mathbf{X}$ is elaborated to $\mathbf{t}_2 = \varepsilon_2(\lambda x : ?_{\mathbf{X}}. \mathbf{t}_1) :: ?_{\mathbf{X}} \rightarrow \mathbf{X}$, where $\varepsilon_2 = \{(?_{\mathbf{X}} \rightarrow \mathbf{X}, \text{inj}_? \rightarrow \text{refl}_{\mathbf{X}}, \text{inj}_? \rightarrow \text{refl}_{\mathbf{X}})\}$ justifies that $?_{\mathbf{X}} \rightarrow \mathbf{X} \sim ?_{\mathbf{X}} \rightarrow \mathbf{X}$.
3. Similarly, the type abstraction is also ascribed: $\lambda \mathbf{X}. \lambda x : ?_{\mathbf{X}}. x :: \mathbf{X}$ elaborates to $\mathbf{t}_3 = \varepsilon_3(\lambda \mathbf{X}. \mathbf{t}_2) :: \forall \mathbf{X}. ?_{\mathbf{X}} \rightarrow \mathbf{X}$, where $\varepsilon_3 = \{(\forall \mathbf{X}. ?_{\mathbf{X}} \rightarrow \mathbf{X}, \forall \mathbf{X}. \text{inj}_? \rightarrow \text{refl}_{\mathbf{X}}, \forall \mathbf{X}. \text{inj}_? \rightarrow \text{refl}_{\mathbf{X}})\}$ justifies that $\forall \mathbf{X}. ?_{\mathbf{X}} \rightarrow \mathbf{X} \sim \forall \mathbf{X}. ?_{\mathbf{X}} \rightarrow \mathbf{X}$.
4. Type application $(\lambda \mathbf{X}. \lambda x : ?_{\mathbf{X}}. x) [\text{Bool}]$ elaborates to $\mathbf{t}_4 = \varepsilon_4(\mathbf{t}_3 [\text{Bool}]) :: ? \rightarrow \text{Bool}$, where $\varepsilon_4 = \{(\text{Bool} \rightarrow \text{Bool}, \text{inj}_{\mathbf{X}} \rightarrow \text{refl}_{\text{Bool}}, \text{inj}_{\text{Bool}} \rightarrow \text{Bool})\}$,

$G \leq G$ Shape-restricted type precision

$$\begin{array}{c}
\leq_B \frac{}{B \leq B} \quad \leq_X \frac{}{X \leq X} \quad \leq_{\rightarrow} \frac{G_1 \leq G_2 \quad G'_1 \leq G'_2}{G_1 \rightarrow G'_1 \leq G_2 \rightarrow G'_2} \quad \leq_{\times} \frac{G_1 \leq G_2 \quad G'_1 \leq G'_2}{G_1 \times G'_1 \leq G_2 \times G'_2} \\
\leq_{\forall} \frac{G_1 \leq G_2}{\forall X. G_1 \leq \forall X. G_2} \quad \leq_{B?} \frac{}{B \leq ?_{\delta}} \quad \leq_{X?} \frac{X \in \delta}{X \leq ?_{\delta}} \quad \leq_{?} \frac{\delta \subseteq \delta'}{?_{\delta} \leq ?_{\delta'}}
\end{array}$$

$\Omega \vdash t : G \sqsubseteq t : G$ Term precision

$$\begin{array}{c}
\sqsubseteq_b \frac{}{\Omega \vdash b : B \sqsubseteq b : B} \quad \sqsubseteq_x \frac{x : G_1 \sqsubseteq G_2 \in \Omega}{\Omega \vdash x : G_1 \sqsubseteq x : G_2} \\
\sqsubseteq_{\lambda} \frac{\Omega, x : G_1 \sqsubseteq G_2 \vdash t_1 : G'_1 \sqsubseteq t_2 : G'_2 \quad G_1 \sqsubseteq G_2}{\Omega \vdash \lambda x : G_1. t_1 : G_1 \rightarrow G'_1 \sqsubseteq \lambda x : G_2. t_2 : G_2 \rightarrow G'_2} \\
\sqsubseteq_{\times} \frac{\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2 \quad \Omega \vdash t'_1 : G'_1 \sqsubseteq t'_2 : G'_2}{\Omega \vdash \langle t_1, t'_1 \rangle : G_1 \times G'_1 \sqsubseteq \langle t_2, t'_2 \rangle : G_2 \times G'_2} \quad \sqsubseteq_{\Lambda} \frac{\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2}{\Omega \vdash \Lambda X. t_1 : \forall X. G_1 \sqsubseteq \Lambda X. t_2 : \forall X. G_2} \\
\sqsubseteq_{\text{asc}} \frac{\Omega \vdash t_1 : G'_1 \sqsubseteq t_2 : G'_2 \quad G_1 \sqsubseteq G_2}{\Omega \vdash t_1 :: G_1 : G_1 \sqsubseteq t_2 :: G_2 : G_2} \quad \sqsubseteq_{\text{app}} \frac{\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2 \quad \Omega \vdash t'_1 : G'_1 \sqsubseteq t'_2 : G'_2}{G_1 \rightarrow G_{11} \rightarrow G_{12} \quad G_2 \rightarrow G_{21} \rightarrow G_{22}}{\Omega \vdash t_1 t'_1 : G_{12} \sqsubseteq t_2 t'_2 : G_{22}} \\
\sqsubseteq_{\text{pairi}} \frac{\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2 \quad G_1 \rightarrow G_{11} \times G_{12} \quad G_2 \rightarrow G_{21} \times G_{22}}{\Omega \vdash \pi_i(t_1) : G_{1i} \sqsubseteq \pi_i(t_2) : G_{2i}} \\
\sqsubseteq_{\text{appG}} \frac{\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2 \quad G_1 \rightarrow \forall X. G'_1 \quad G_2 \rightarrow \forall X. G'_2 \quad G_1 \leq G_2}{\Omega \vdash t_1 [F] : G'_1[F/X] \setminus X \sqsubseteq t_2 [F] : G'_2[F/X] \setminus X}
\end{array}$$

Figure 7.8: $F^?$: Shape-restricted type precision and term precision.

$(? \rightarrow \mathbf{Bool}, \text{inj}_{? \rightarrow \text{refl}_{\mathbf{Bool}}}, \text{inj}_{? \rightarrow \text{refl}_{\mathbf{Bool}}})$ justifies that $?_X \rightarrow \mathbf{Bool} \sim ? \rightarrow \mathbf{Bool}$ (sealing/unsealing evidence).

5. Number 42 is ascribed to its type and elaborated to $t_5 = \varepsilon_5 \mathbf{42} :: \mathbf{Int}$, where ε_5 justifies the reflexive judgment $\mathbf{Int} \sim \mathbf{Int}$ and $\varepsilon_5 = \{(\mathbf{Int}, \text{refl}_{\mathbf{Int}}, \mathbf{Int})\}$.
6. The elaborated function argument t_5 is ascribed to the domain of the function type: $t_6 = \varepsilon_6 t_5 :: ?$, where $\varepsilon_6 = \{(\mathbf{Int}, \text{refl}_{\mathbf{Int}}, \text{inj}_{\mathbf{Int}})\}$ which justifies that $\mathbf{Int} \sim ?$.
7. Finally, the whole program elaborates to $t_4 t_6$:
 $(\varepsilon_4 (\varepsilon_3 (\Lambda X. \varepsilon_2 (\lambda x : ?_X. \varepsilon_1 x :: X) :: ?_X \rightarrow X) :: \forall X. ?_X \rightarrow X [\mathbf{Bool}]) :: ? \rightarrow \mathbf{Bool}) (\varepsilon_6 (\varepsilon_5 \mathbf{42} :: \mathbf{Int}) :: ?)$.

7.3 Source-level graduality

Under the natural notion of type precision (Figure 7.4), some $F^?$ terms related by precision elaborate to $F_\varepsilon^?$ terms that are *not* related by precision. Consider program $(\lambda X.\lambda x:X.x) :: \forall X.?X \rightarrow ?X$ [Int] 1 more precise than $(\lambda X.\lambda x:X.x) :: \forall X.?X$ [Int] 1 (note that $\forall X.?X \rightarrow ?X \sqsubseteq \forall X.?X$). The first program elaborates to a program that reduces correctly, but the second to a program that fails. As explained in the previous section, this is because $\text{instEv}(?X, X, \text{Int})$ does not generate evidence that contains function spans that seal the argument. This does not mean that there is no source-level graduality in $F^?$ at all; as first explored by Igarashi et al. (2017a), the fact that the gradual guarantees are stated relative to a notion of precision means that we may be able to characterize source-level graduality via a restricted notion of precision.

To characterize the $F^?$ programs for which we can guarantee graduality, it is enough to restrict term precision *only* for type applications, enforcing that for such expressions, type precision be restricted to types of the same shape. Notice how rule ($\sqsubseteq_{\text{appG}}$) in Figure B.8 uses *shape-restricted* type precision \leq in its premise, while other rules, such as rule (\sqsubseteq_{asc}), use the natural type precision \sqsubseteq . Shape-restricted type precision \leq is defined similarly to \sqsubseteq , but in the case of polymorphic and function types, the type constructors have to match. For example, $\forall X.X \rightarrow X \leq \forall X.?X \rightarrow ?X$ but $\forall X.(X \rightarrow X) \rightarrow X \not\leq \forall X.?X \rightarrow ?X$ and $\forall X.X \rightarrow X \not\leq \forall X.?X$. This means that if $G_1 \leq G_2$, then all sealing spans included in applying instEv to G_1 will be included in the application of instEv to G_2 . The other cases of term precision are derived just as the natural lifting of type precision to terms.

With this notion of term precision, two source terms related by precision elaborate to $F_\varepsilon^?$ terms that are also related by precision:

Lemma 7.10 *If $\vdash t_1 : G_1 \sqsubseteq t_2 : G_2$, $\vdash t_1 : G_1 \rightsquigarrow t_1$ and $\vdash t_2 : G_2 \rightsquigarrow t_2$, then $\vdash \text{initPT}(G_1, G_2) : t_1 \sqsubseteq t_2$.*

Note that precision in $F_\varepsilon^?$ is proof-relevant, therefore we have to provide a proof term that justifies “how” two terms are related. We do that by using the initial proof term function between the types of the related terms.

The dynamic semantics of a $F^?$ term are given by first elaborating the term to $F_\varepsilon^?$ and then reducing the $F_\varepsilon^?$ term. Hence, for establishing the gradual guarantees in $F^?$, we write $t \Downarrow v$ if $\vdash t : G \rightsquigarrow t$ and $t \xrightarrow{*} v$. Similarly, we write $t \Uparrow$ if the elaboration of t diverges. Then, using Lemmas 7.9, 7.10 and 6.14 we can prove the gradual guarantees for $F^?$:

Theorem 7.11 (Gradual guarantees) *Suppose $\vdash t_1 : G_1 \sqsubseteq t_2 : G_2$ and $\vdash t_1 : G_1$.*

1. $\vdash t_2 : G_2$ and $G_1 \sqsubseteq G_2$.
2. If $t_1 \Downarrow v_1$, then $t_2 \Downarrow v_2$ and $\vdash \text{initPT}(G_1, G_2) : v_1 \sqsubseteq v_2$.
If $t_1 \Uparrow$ then $t_2 \Uparrow$.

7.4 Source-level parametric reasoning

As a first form of parametric reasoning for $F^?$, the elaborations of well-typed $F^?$ terms produces $F_\varepsilon^?$ terms that are also well typed (by Theorem 7.9), and hence related to themselves (by Theorem 6.8):

Corollary 7.12 *If $\Delta; \Gamma \vdash t : G \rightsquigarrow t$ then $\Delta; \Gamma \vdash t \preceq t : G$.*

This lemma is powerful, but it is not immediately clear what it means for concrete example terms in $F^?$. We make this clearer as follows: a type abstraction f of type $\forall X.G$ applied to related types, produces related terms whenever X does not occur in the scopes of unknown types in G (a condition written $G \setminus X = G$ below):

Lemma 7.13 $\forall n, \rho$

$$\frac{\Delta; \Gamma \vdash f : \forall X.G \quad \forall B_1, B_2, R \in \text{REL}[B_1, B_2] \quad ((n, \rho) \in \mathcal{D}[\Delta] \wedge (n, \gamma) \in \mathcal{G}_\rho[\Gamma]) \quad G \setminus X = G \quad \Delta; \Gamma \vdash f [B_i] : G[B_i/X] \setminus X \rightsquigarrow t_i}{(n, \rho_1(\gamma_1(t_1)), \rho_2(\gamma_2(t_2))) \in \mathcal{J}_{\rho, X \mapsto (B_1, B_2, R)}[\mathbf{G}]}$$

As a direct consequence of Lemma 7.13, every $F^?$ program ascribed to a static type behaves parametrically, even if it internally uses the unknown type. Since the logical relation for $F^?$ coincides almost exactly with traditional formulations for System F, this means we get the same reasoning about such applications than in System F itself, save for the possibility that two related terms both raise a runtime error. From Lemma 7.13, we can also derive free theorems involving imprecise types. For instance, given a function of type $\forall X.?_\emptyset \rightarrow X$, then the application of the function either fails or diverges.

Lemma 7.14 *If $\vdash f : \forall X.?_\emptyset \rightarrow X$, $\vdash v : B$ and $t = f [B] v$, then $t \Downarrow \mathbf{error}$, or $t \Uparrow$.*

This behavior is expected because $?_\emptyset$ can only stand for a type that does *not* involve X . Intuitively, this means that this gradual polymorphic function type denotes types such as $\forall X.\text{Int} \rightarrow X$ and $\forall X.\text{Bool} \rightarrow X$, but not $\forall X.X \rightarrow X$. Therefore, the function cannot create a value of type X out of thin air, and the argument v cannot possibly be sealed as a value of type X , so the function necessarily fails if it tries to return any value.

Lemma 7.13 does not apply to polymorphic functions whose type mentions unknown types with the quantified variable in scope. To understand why such types require more nuance, remember the example function $f = \Lambda X. \lambda x : ?x :: X$ discussed in the introduction. When applied to type `Int` and value `42`, this function produces the value `42`, but it throws a runtime type error when applied to `Bool` and value `42`. In such examples, $F^?$ inserts plausible sealing evidence, in an effort to guess whether the programmer intended `42` to be treated as a value of type X or not, in a maximally permissive way. However, this does not mean we do not get any form of parametricity for such examples, but rather, we need to keep in mind the intuitive effect of plausible sealing. In other words, $F^?$ supports more parametric reasoning than just what Lemma 7.13 expresses. Particularly, when a type variable is in the scope of an unknown type within a function type, we can also derive some free theorems using Corollary 7.12. For instance, if a function f has type $\forall X.?_X \rightarrow X$, then by parametricity we can deduce that f behaves either as the identity function or fails or diverges:

Lemma 7.15 *If $\vdash f : \forall X.?_X \rightarrow X$, $\vdash v : B$ and $t = f [B] v$, then $t \Downarrow v$ with $\vdash v : B \rightsquigarrow v$, or $t \Downarrow \mathbf{error}$, or $t \Uparrow$.*

Contrary to GSF, in $F^?$ this result holds just by looking at the type of f , without the need to unfold its definition. Intuitively, this lemma takes into account that $F^?$ applies plausible sealing to the argument v , so f might return it as the result of type X . The function f can also diverge or fail,

but parametricity for $F^?$ still implies that f cannot return any value other than v .

Finally, we establish that if a function has type $\forall X. \forall Y. ?_{X,Y} \rightarrow X \times Y$, i.e. it receives one argument but returns a pair, then if instantiated with the same type and applied with a value of the given type, it returns a pair of the duplicated value.

Lemma 7.16 *If $\vdash f : \forall X. \forall Y. ?_{X,Y} \rightarrow X \times Y$, $\vdash v : B$ and $t = f [B] [B] v$, then $t \Downarrow v$ and $\langle v, v \rangle \Downarrow v$, or $t \Downarrow \mathbf{error}$, or $t \Uparrow$.*

Otherwise, if the function is instantiated to different types, then the application should fail or diverge, as it cannot create a value of type Y (or X).

Lemma 7.17 *If $\vdash f : \forall X. \forall Y. ?_{X,Y} \rightarrow X \times Y$, $B_1 \neq B_2$, $\vdash v : B_1$ and $t = f [B_1] [B_2] v$, then $t \Downarrow \mathbf{error}$, or $t \Uparrow$.*

Fully-abstract embedding? Jacobs et al. (2021) propose the fully-abstract embedding of a statically-typed language into its gradually-typed counterpart as a criteria to assess gradual languages. Informally, this means that terms contextually equivalent in the static language must also be contextually equivalent in the gradual language. Such a result has been disproved for λB via the following counterexample (Devriese et al., 2018). Consider type $\mathbf{Univ} \triangleq \exists Y. \forall X. (X \rightarrow Y) \times (Y \rightarrow X)$ and the next two terms:

$$\begin{aligned} t_s &\triangleq \lambda x : \mathbf{Univ}. \text{unpack } x \text{ as } \langle Y, x' \rangle \text{ in let } x'' : (\mathbf{Unit} \rightarrow Y) \times (Y \rightarrow \mathbf{Unit}) = x' [\mathbf{Unit}] \text{ in } \pi_2(x'') (\pi_1(x'') \text{ unit}) \\ t_u &\triangleq \lambda x : \mathbf{Univ}. \text{unpack } x \text{ as } \langle Y, x' \rangle \text{ in let } x'' : (\mathbf{Unit} \rightarrow Y) \times (Y \rightarrow \mathbf{Unit}) = x' [\mathbf{Unit}] \text{ in } \pi_2(x'') (\pi_1(x'') \text{ unit}); \omega_{\mathbf{Unit}} \end{aligned}$$

where $\omega_{\mathbf{Unit}}$ is the always-diverging term of type \mathbf{Unit} . Intuitively, t_s and t_u are almost the same function, but the $\omega_{\mathbf{Unit}}$ in t_u makes it always diverge when invoked. Devriese et al. prove that the invocation $\pi_2(x'') (\pi_1(x'') \text{ unit})$ must diverge for all System F inhabitants of \mathbf{Univ} , so that the terms t_s and t_u are contextually equivalent.

In λB (and GSF) these two terms are no longer contextually equivalent: there is a context in which the application of t_s terminates. Essentially, the globally scoped sealing used in these languages allows constructing an inhabitant of \mathbf{Univ} which instantiates Y to the unknown type and does not make the term t_s diverge.

While proving fully-abstract embedding for $F_\epsilon^?$ is future work, we can already show that the lexically scoped sealing of $F_\epsilon^?$ breaks this counterexample. The following lemma states that the elaborations of t_s and t_u are related.

Lemma 7.18 *If $\vdash t_s : \mathbf{Univ} \rightarrow \mathbf{Unit} \rightsquigarrow t_s$ and $\vdash t_u : \mathbf{Univ} \rightarrow \mathbf{Unit} \rightsquigarrow t_u$, then $\vdash t_s \preceq t_u : \mathbf{Univ} \rightarrow \mathbf{Unit}$.*

In the appendix we sketch a proof of this lemma encoding existential types in terms of universal types (Harper, 2012).

7.5 Discussion and Related Work

Limitations. The technical development of plausible sealing in this article suffers from two technical limitations. First, we only formalize a simplified form of polymorphism restricted to instantiations with base and variable types. Second, graduality for $F^?$ is restricted for type applications, since two type applications are only related when the polymorphic types have the same shape (Section 7).

Both limitations manifest in the definition of instantiation evidence for the unknown type (Figure 7.7): $\text{instEv}(\delta_{\delta_1;X;\delta_2}, X, F) = \{(\delta_{\delta_1;\delta_2}, \text{inj}_?, \text{inj}_?), (F, \text{inj}_X, \text{inj}_F)\}$. Recall that the role of this instantiation evidence is to cast, for example, a function application of type $?_{X:\text{Int}} \rightarrow \text{Int}$ to type $?_{\emptyset} \rightarrow \text{Int}$. The restricted form of polymorphism is apparent because instEv 's third argument F is restricted to an instantiation type (a base type or a type variable), and we simply use $\text{inj}_F : F \sqsubseteq \delta_{\delta_1;\delta_2}$ to inject F into type $\delta_{\delta_1;\delta_2}$. Generalizing to full polymorphism would require replacing F and inj_F in the above definition with an arbitrary type G and a proof term $c : G \sqsubseteq \delta_{\delta_1;\delta_2}$. This requires to extend the syntax of δ to allow for any type. An initial exploration suggests this is largely unproblematic and in fact, our Agda proofs of consistent transitivity associativity and monotonicity already support such a richer syntax of δ . A problem turns up when one of the other types in δ_1 or δ_2 mentions $?_{\delta'}$ with $X : X \in \delta'$. In that case, it appears we additionally need a proof term that expresses precision between $?_{\delta}$ and $?_{\delta'}$ when δ is not just a subset of δ' but some of the types in δ are themselves strictly more precise than corresponding types in δ' .

Allowing a type abstraction to be instantiated at any type G would require saving the information of the instantiated type in the proof term inj_X . This information could then be refined through composition. Therefore, we should transform inj_X to the proof term sequence $\text{inj}_X(c)$, where $\text{inj}_X(c) : G' \sqsubseteq \delta$, $X : G \in \delta$, $c : G' \sqsubseteq G$ and $\delta \vdash G'$. In addition, the composition of proof terms for this case would slightly change: $c_1; \text{inj}_X(c_2) = \text{inj}_X(c_1; c_2)$. Note that applying the type abstraction to any type G' with rule RappG would not change; the instantiated type G' would continue to be substituted in the body of the type abstraction and the evidence.

The other main limitation of $F^?$ is caused by the right-hand-side of the above definition. Problematically, it only mentions two cases: a value of type $?_{\delta_1;\delta_2}$ will be converted into type $?_{\delta_1;X;\delta_2}$ either (1) by not sealing at all and simply extending the scope of the unknown type, or (2) by sealing, converting a value of type F into a value sealed at type X in $?_{\delta_1;X;\delta_2}$. What is missing is a recursive case that would treat, for example, a value of type $F \rightarrow F$ as a value of type $X \rightarrow X$ and recursively seal it accordingly. In fact, this could be accommodated easily by extending the right-hand-side with an additional case: $(F \rightarrow F, \text{inj}_{\rightarrow}(\text{inj}_X \rightarrow \text{inj}_X), \text{inj}_{\rightarrow}(\text{inj}_F \rightarrow \text{inj}_F))$. Because there are an infinite amount of such cases, we conjecture that a solution is to introduce a syntax of recursive evidence that would allow us to define $\text{instEv}(\delta_{\delta_1;X;\delta_2}, X, F)$ as:

$\mu\varepsilon. \text{inj}_{\rightarrow}(\varepsilon \rightarrow \varepsilon) \uplus \text{inj}_{\times}(\varepsilon \times \varepsilon) \uplus \text{inj}_{\forall}(\forall Y. \varepsilon) \uplus \{(B, \text{inj}_B, \text{inj}_B)\} \uplus \{(F, \text{inj}_X, \text{inj}_F)\}$. In this notation, we construct an evidence $\text{inj}_{\rightarrow}(\varepsilon \rightarrow \varepsilon) : ?_{\delta} \sim ?_{\delta'}$ from $\varepsilon : ?_{\delta} \sim ?_{\delta'}$ by combining two spans (G_1, c_1, c_2) and (G_2, c_3, c_4) from ε into the span:

$$(G_1 \rightarrow G_2, \text{inj}_{\rightarrow}(c_1 \rightarrow c_3), \text{inj}_{\rightarrow}(c_2 \rightarrow c_4))$$

and similarly for inj_{\times} and inj_{\forall} . We leave the definition of the operational behavior of such recursive evidence and the proofs of its properties to future work.

Related Work. *Strict Precision.* Igarashi et al. (2017a) first proposed using a non-standard notion of precision in System F_G to address some problems with the dynamic gradual guarantee when the unknown type is allowed to stand for a type variable. Consequently, in System F_G the unknown

type is not consistent with any type variable. Here, the source language $F^?$ also restricts precision, but in a much less drastic manner: type precision is the standard precision, and only term precision is restricted, in order to only relate type applications to types of the same shape. We explain in the previous section how this restriction could be lifted. Finally, $F_\varepsilon^?$ has no such restriction, and satisfies the gradual guarantees with respect to the standard notion of precision, for both types and terms.

Family of Unknown Types. Devriese et al. (2018) proposed to decorate the unknown type with the set of type variables in scope, as we do for plausible sealing, thus limiting the expressiveness of the unknown type by forming different families. This proposal aims to potentially reestablish the fully abstract embedding property of System F into λB . Devriese et al. (2018) and more recently Jacobs et al. (2021) proposed a new criterion for gradual typing named the fully abstract embedding (FAE) property: the embedding from the static to the gradual language should be fully abstract in order to preserve the semantics properties of the static languages. We conjecture that this criterion holds for the language $F^?$, being a gradual version of System F that preserves its main semantic property (*i.e.* parametricity), although we leave a proof of FAE as future work.

Implicit Polymorphism. Several polymorphic gradual languages have explored implicit polymorphism present in languages such as Haskell. Xie et al. (2018) developed a gradual source language with implicit polymorphism, where the runtime semantics are given by compilation to λB . λB and System F_G , in turn, are languages with explicit polymorphism that accommodate some form of implicit polymorphism. $F^?$ and $F_\varepsilon^?$, as well as PolyG ν and GSF, only support explicit polymorphism; exploring implicit polymorphism for $F^?$ is an interesting venue for future work, in order to enhance interoperability between typed and untyped code.

7.6 Conclusion

This chapter presents the source language $F^?$ and its elaboration to $F_\varepsilon^?$, illustrating the practicality of $F_\varepsilon^?$ as a target language for a type-driving source language. The elaboration from $F^?$ to $F_\varepsilon^?$ is novel, inserting maybe-sealing forms statically. We establish graduality for $F^?$ subject to a restriction on type applications and explain the source-level parametric reasoning that $F^?$ offers. We conjecture, and leave as future work, that by representing the evidence of instantiation recursively when imprecise types occur, we can lift the restriction on the term precision relation, thus correctly running all programs that by graduality must end in a value.

Chapter 8

Conclusion

In this thesis, we have studied the integration of parametric polymorphism and gradual typing into a language while preserving parametricity, graduality, and the familiar syntax of System F. We uncover design flaws in prior work on gradual parametric languages that enforce relational parametricity. We exploit the Abstracting Gradual Typing (AGT) methodology to design a new gradual language with explicit parametric polymorphism, GSF. We find that AGT largely simplifies the static semantics of GSF but does not produce a language that respects parametricity; nontrivial exploration was necessary to discover how to design the evidence and the consistent transitive operator in order to recover parametricity. GSF goes beyond prior work based on the type-driven sealing approach, archiving a unique combination of properties like harmless imprecise ascriptions, expressiveness of imprecision, type instantiations faithful, and the embedding of the untyped lambda calculus with sealing/unsealing primitives. Although GSF lacks complete support for the dynamic gradual guarantee, we precisely characterize the static-to-dynamic checking spectrum that it supports, allowing the programmer to reason about graduality.

As a result of the study and development of GSF, we found that the tension between the gradual guarantees and parametricity in the type-driven approach comes from the early commitment to seal values based on type information. Inspired by this observation, we proposed plausible sealing as a new intermediate language mechanism that postpones such decisions to runtime. We developed $F_\epsilon^?$, an intermediate language with plausible sealing that fully satisfies both parametricity and graduality in a simplified setting. An outstanding feature of $F_\epsilon^?$ is that it avoids dynamic global sealing, which has been the only mechanism for gradual parametricity used so far. In this regard, we propose a novel lexically-scoped form of sealing realized using a representation of evidence inspired by the category of spans. Consequently, $F_\epsilon^?$ satisfies a standard formulation of parametricity that does not break System F equivalences. $F_\epsilon^?$ was designed to serve as an intermediate language for a source language without explicit sealing, *i.e.*, a language base on System F syntax. Hence, in order to show the practicality of plausible sealing, we describe a translation from $F^?$, a source language, to $F_\epsilon^?$, that takes care of inserting plausible sealing forms. We establish graduality for $F^?$, subject to a restriction on type applications, and explain the source-level parametric reasoning it supports. Although the results presented in this work still have some restrictions, they open a new path towards the goal of reconciling parametricity and graduality in a language with System F syntax. Moreover, some of our novel techniques are potentially reusable in other environments, which would be very promising to study.

The efficient implementation of gradual typing in programming languages while preserving

soundness guarantees is still an open research area. The few existing implementations suffer from performance overheads due to the inserted runtime checks to preserve type soundness (Takikawa et al., 2016). Therefore, whether the theory developed in this thesis for gradual parametricity can be practically applied to languages remains an open question. This thesis focuses on the semantics and metatheoretical properties of gradual parametric polymorphism without taking into account efficiency considerations such as pay-as-you-go (Siek and Taha, 2006; Igarashi et al., 2017a), space efficiency (Herman et al., 2010; Siek and Wadler, 2010), cast elimination (Rastogi et al., 2012), etc. Optimizing the dynamic semantics of GSF and $F_\epsilon^?$ is left for future work. Likewise, we leave blame tracking as future work, which is valuable to report more informative error messages, but most importantly, to identify error cases properly. We expect ongoing work by colleagues on incorporating blame into AGT to be directly applicable to GSF and $F_\epsilon^?$ because there do not seem to be any parametricity-specific challenges related to blame. The use of explicit polymorphism in the design of GSF and $F_\epsilon^?$ hampers certain interoperability scenarios. Hence, it would be very valuable to investigate the resolution of this tension. It would also be interesting to extend gradual parametricity with other widely used features in programming languages, such as references and recursive types, to study the interaction and metatheory of gradual parametric polymorphism in these settings. In addition, it would be worth studying whether $F_\epsilon^?$ satisfies the ambitious criterion to assess gradual languages proposed by Jacobs et al. (2021): fully abstract embedding of the statically-typed language into the gradually-typed language. So far, this criterion has not been proven for any of the gradual parametric languages proposed in the literature.

As extensively discussed, gradual parametricity is subtle, and there are many scenarios when the decision of failing or not is open to debate and various considerations. This thesis contributes to this discussion by proposing several practical principles, such as ensuring that fully-static terms can be embedded in gradual contexts and made imprecise externally without affecting their behavior. As we pointed out previously in Chapter 7, a strict interpretation of source-level parametricity is incompatible with graduality. We can find a pair of source programs that, by parametricity, both should fail, but by graduality, they should reduce to a value $((\Lambda X.\lambda x :?.x :: X) [\text{Int}] 42$ and $(\Lambda X.\lambda x :?.x :: X) [\text{Bool}] 42)$. We decided to preserve graduality, admitting some non-parametric behaviors in the design of the source language $F^?$. However, we consider these non-parametric behaviors acceptable, allowing us to continue reasoning about parametric types. First, uniformity of behavior is satisfied for polymorphic functions of fully precise types, even if they internally use imprecise types. In other words, the differences in behavior can only occur for imprecise types (and can therefore be avoided using ascriptions to precise types). Second, the behavior of a parametric program is entirely predictable based on type information available statically at the call site and does not depend on runtime type information. In order to preserve graduality and the strict interpretation of parametricity in a language, it is necessary to introduce an explicit sealing/unsealing mechanism, as is the case with PolyG and $F_\epsilon^?$. The difference between $F_\epsilon^?$ and PolyG $^\nu$ is that the former, besides allowing explicitly declaring sealing/unsealing expressions, also allows maybe-sealing expressions. These maybe-sealing expressions, or what is the same plausible sealing, allow $F_\epsilon^?$ to become the target language of a source gradual parametric language.

Acknowledgments

This work was partially supported by the United States Air Force Office of Scientific Research under award number FA9550-21-1-0054, by the Flemish Research Programme Cybersecurity, and by ANID FONDECYT projects 1190058 and 3200583, Chile.

Bibliography

- Martin Abadi, Luca Cardelli, Benjamin Pierce, and Gordon Plotkin. 1991. Dynamic typing in a statically typed language. *ACM Transactions on Programming Languages and Systems* 13, 2 (April 1991), 237–268.
- Martin Abadi, Luca Cardelli, Benjamin Pierce, and Didier Rémy. 1995. Dynamic typing in polymorphic languages. *Journal of Functional Programming* 5, 1 (1995), 111–130.
- Amal Ahmed. 2004. *Semantics of Types for Mutable State*. Ph.D. Dissertation. Princeton University.
- Amal Ahmed. 2006. Step-Indexed Syntactic Logical Relations for Recursive and Quantified Types. In *Proceedings of the 15th European Symposium on Programming Languages and Systems (ESOP 2006) (Lecture Notes in Computer Science)*, Peter Sestoft (Ed.), Vol. 3924. Springer-Verlag, Vienna, Austria, 69–83.
- Amal Ahmed, Derek Dreyer, and Andreas Rossberg. 2009a. State-dependent representation independence. In *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2009)*. ACM Press, Savannah, GA, USA, 340–353.
- Amal Ahmed, Robert Bruce Findler, Jacob Matthews, and Philip Wadler. 2009b. Blame for All. In *Workshop on Script to Program Evolution (STOP)*. Genova, Italy.
- Amal Ahmed, Robert Bruce Findler, Jeremy G. Siek, and Philip Wadler. 2011. Blame for All. In *Proceedings of the 38th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2011)*. ACM Press, Austin, Texas, USA, 201–214.
- Amal Ahmed, Dustin Jamner, Jeremy G. Siek, and Philip Wadler. 2017. Theorems for Free for Free: Parametricity, with and Without Types. *SeeICFP 2017 (2017)*, 39:1–39:28.
- Andrew W. Appel and David McAllester. 2001. An Indexed Model of Recursive Types for Foundational Proof-carrying Code. *ACM Transactions on Programming Languages and Systems* 23, 5 (Sept. 2001), 657–683.
- Arthur Azevedo de Amorim, Matt Fredrikson, and Limin Jia. 2020. Reconciling Noninterference and Gradual Typing. In *Proceedings of the 2020 Symposium on Logic in Computer Science (LICS 2020)*.
- Johannes Bader, Jonathan Aldrich, and Éric Tanter. 2018. Gradual Program Verification. In *Proceedings of the 19th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2018) (Lecture Notes in Computer Science)*, Işıl Dillig and Jens Palsberg (Eds.), Vol. 10747. Springer-Verlag, Los Angeles, CA, USA, 25–46.
- Felipe Bañados Schwerter, Alison M. Clark, and Jafery. 2021. Abstracting Gradual Typing Moving Forward: Precise and Space-Efficient. *SeePOPL 2021 (2021)*, 61:1–61:28.

- Felipe Bañados Schwerter, Ronald Garcia, and Éric Tanter. 2014. A Theory of Gradual Effect Systems. In *Proceedings of the 19th ACM SIGPLAN Conference on Functional Programming (ICFP 2014)*. ACM Press, Gothenburg, Sweden, 283–295.
- Felipe Bañados Schwerter, Ronald Garcia, and Éric Tanter. 2016. Gradual Type-and-Effect Systems. *Journal of Functional Programming* 26 (Sept. 2016), 19:1–19:69.
- Gavin Bierman, Erik Meijer, and Mads Torgersen. 2010. Adding Dynamic Types to C#. In *Proceedings of the 24th European Conference on Object-oriented Programming (ECOOP 2010) (Lecture Notes in Computer Science)*, Theo D’Hondt (Ed.). Springer-Verlag, Maribor, Slovenia, 76–100.
- Rastislav Bodík and Rupak Majumdar (Eds.). 2016. *Proceedings of the 43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2016)*. ACM Press, St Petersburg, FL, USA.
- Robert Cartwright and Mike Fagan. 1991. Soft typing. In *Proceedings of the ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI)*. Toronto, Ontario, Canada, 278–292.
- Giuseppe Castagna and Victor Lanvin. 2017. Gradual Typing with Union and Intersection Types. *SeeICFP 2017* (2017), 41:1–41:28.
- Matteo Cimini and Jeremy Siek. 2016. The gradualizer: a methodology and algorithm for generating gradual type systems, See Bodík and Majumdar (2016), 443–455.
- Michael R. Clarkson and Fred B. Schneider. 2008. Hyperproperties. In *2008 21st IEEE Computer Security Foundations Symposium*. 51–65. <https://doi.org/10.1109/CSF.2008.7>
- Haskell B. Curry, J. Roger Hindley, and J. P. Seldin. 1972. *Combinatory Logic, Volume II*. Studies in logic and the foundations of mathematics, Vol. 65. North-Holland Pub. Co.
- Luis Damas and Robin Milner. 1982. Principal Type-Schemes for Functional Programs. In *Proceedings of the 9th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 1982)*. ACM Press, New York, NY, USA, 207–212. <https://doi.org/10.1145/582153.582176>
- Dominique Devriese, Marco Patrignani, and Frank Piessens. 2018. Parametricity versus the universal type. *Proceedings of the ACM on Programming Languages* 2, POPL (Jan. 2018), 38:1–38:23.
- Tim Disney and Cormac Flanagan. 2011. Gradual information flow typing. In *International Workshop on Scripts to Programs*.
- Joseph Eremondi, Éric Tanter, and Ronald Garcia. 2019. Approximate Normalization for Gradual Dependent Types. *Proceedings of the ACM on Programming Languages* 3, ICFP (Aug. 2019), 88:1–88:30.
- Luminous Fennell and Peter Thiemann. 2013. Gradual Security Typing with References. In *Proceedings of the 26th Computer Security Foundations Symposium (CSF)*. 224–239.
- Luminous Fennell and Peter Thiemann. 2016. LJGS: Gradual Security Types for Object-Oriented Languages. In *Proceedings of the 30th European Conference on Object-Oriented Programming (ECOOP 2016) (Leibniz International Proceedings in Informatics (LIPIcs))*, Shriram Krishnamurthi and Benjamin S. Lerner (Eds.), Vol. 56. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Rome, Italy, 9:1–9:26.

- Ronald Garcia, Alison M. Clark, and Éric Tanter. 2016. Abstracting Gradual Typing, See Bodík and Majumdar (2016), 429–442. See erratum: <https://www.cs.ubc.ca/rxg/agt-erratum.pdf>.
- Ronald Garcia and Éric Tanter. 2015. Deriving a Simple Gradual Security Language. eprint arXiv:1511.01399.
- Ronald Garcia, Éric Tanter, Roger Wolff, and Jonathan Aldrich. 2014. Foundations of Typestate-Oriented Programming. *ACM Transactions on Programming Languages and Systems* 36, 4, Article 12 (Oct. 2014), 12:1–12:44 pages.
- Jean-Yves Girard. 1972. *Interprétation Fonctionnelle et Élimination des Coupures de l'Arithmétique d'Ordre Supérieur*. Ph.D. Dissertation. Université de Paris VII, Paris, France.
- Arjun Guha, Jacob Matthews, Robert Bruce Findler, and Shriram Krishnamurthi. 2007. Relationally-parametric polymorphic contracts. In *Proceedings of the ACM Dynamic Languages Symposium (DLS 2007)*. ACM Press, Montreal, Canada, 29–40.
- Robert Harper. 2012. *Practical Foundations for Programming Languages*. Cambridge University Press.
- David Herman, Aaron Tomb, and Cormac Flanagan. 2010. Space-efficient gradual typing. *Higher-Order and Symbolic Computation* 23, 2 (June 2010), 167–189.
- Kuen-Bang Hou, Nick Benton, and Robert Harper. 2016. Correctness of Compiling Polymorphism to Dynamic Typing. *Journal of Functional Programming* 27 (2016), 1:1–1:24.
- ICFP 2017 2017.
- Atsushi Igarashi, Peter Thiemann, Vasco T. Vasconcelos, and Philip Wadler. 2017b. Gradual Session Types. See ICFP 2017 (2017), 38:1–38:28.
- Yuu Igarashi, Taro Sekiyama, and Atsushi Igarashi. 2017a. On Polymorphic Gradual Typing. See ICFP 2017 (2017), 40:1–40:29.
- Lintaro Ina and Atsushi Igarashi. 2011. Gradual typing for generics. In *Proceedings of the 26th ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA 2011)*. ACM Press, Portland, Oregon, USA, 609–624.
- Koen Jacobs, Amin Timany, and Dominique Devriese. 2021. Fully abstract from static to gradual. See POPL 2021 (2021), 7:1–7:30.
- Elizabeth Labrada, Matías Toro, Éric Tanter, and Dominique Devriese. 2022. Plausible Sealing for Gradual Parametricity. *Proceedings of the ACM on Programming Languages* 6, OOPSLA1 (April 2022), 70:1–70:28.
- Nico Lehmann and Éric Tanter. 2017. Gradual Refinement Types. In *Proceedings of the 44th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2017)*. ACM Press, Paris, France, 775–788.
- Xavier Leroy and Michel Mauny. 1991. Dynamics in ML. In *Proceedings of the Conference on Functional Programming Languages and Computer Architecture (FPCA 1991) (Lecture Notes in Computer Science)*, Vol. 523. Springer-Verlag, 406–426.

- Paul Blain Levy. 1999. Call-by-Push-Value: A Subsuming Paradigm. In *4th International Conference on Typed Lambda Calculi and Applications (TLCA'99) (Lecture Notes in Computer Science)*, Jean-Yves Girard (Ed.), Vol. 1581. Springer-Verlag, 228–242.
- Jacob Matthews and Amal Ahmed. 2008. Parametric Polymorphism Through Run-Time Sealing, or, Theorems for Low, Low Prices!. In *Proceedings of the 17th European Symposium on Programming Languages and Systems (ESOP 2008) (Lecture Notes in Computer Science)*, Sophia Drossopoulou (Ed.), Vol. 4960. Springer-Verlag, Budapest, Hungary, 16–31.
- Jacob Matthews and Robert Bruce Findler. 2007. Operational Semantics for Multi-language programs. In *Proceedings of the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2007)*. ACM Press, Nice, France, 3–10.
- John C. Mitchell. 1988. Polymorphic Type Inference and Containment. *Information and Computation* 76, 2-3 (Feb. 1988), 211–249.
- John C. Mitchell and Gordon D. Plotkin. 1988. Abstract Types Have Existential Type. *ACM Transactions on Programming Languages and Systems* 10, 3 (July 1988), 470–502. <https://doi.org/10.1145/44501.45065>
- James H. Morris. 1973. Protection in Programming Languages. *Commun. ACM* 16, 1 (Jan. 1973), 15–21.
- Georg Neis, Derek Dryer, and Andreas Rossberg. 2009. Non-Parametric Parametricity. In *Proceedings of the 14th ACM SIGPLAN Conference on Functional Programming (ICFP 2009)*. ACM Press, Edinburgh, Scotland, UK, 135–148.
- Max S. New and Amal Ahmed. 2018. Graduality from Embedding-Projection Pairs. , 73:1–73:30 pages.
- Max S. New, Dustin Jamner, and Amal Ahmed. 2020. Graduality and Parametricity: Together Again for the First Time. *Proceedings of the ACM on Programming Languages* 4, POPL (Jan. 2020), 46:1–46:32.
- nLab contributors. 2021a. pullback. <https://ncatlab.org/nlab/show/pullback>
- nLab contributors. 2021b. span. <https://ncatlab.org/nlab/show/span>
- Martin Odersky and Konstantin Läuffer. 1996. Putting Type Annotations to Work. In *Proceedings of the 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 96)*. ACM Press, St. Petersburg Beach, Florida, USA, 54–67.
- Oxford. 2021. *Oxford Advanced Learner's Dictionary* (10th ed.). Oxford University Press. Evidence.
- Benjamin Pierce and Eijiro Sumii. 2000. Relating Cryptography and Polymorphism. Manuscript.
- Benjamin C. Pierce. 2002. *Types and programming languages*. MIT Press, Cambridge, MA, USA.
- POPL 2021 2021.
- Aseem Rastogi, Avik Chaudhuri, and Basil Hosmer. 2012. The ins and outs of gradual type inference. In *Proceedings of the 39th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2012)*. ACM Press, Philadelphia, USA, 481–494.

- John C. Reynolds. 1974. Towards a Theory of Type Structure. In *Proceedings of the Programming Symposium (Lecture Notes in Computer Science)*, Vol. 19. Springer-Verlag, 408–423.
- John C. Reynolds. 1983. Types, abstraction, and parametric polymorphism. In *Information Processing 83*, R. E. A. Mason (Ed.). Elsevier, 513–523.
- Andreas Rossberg. 2003. Generativity and dynamic opacity for abstract types. In *Proceedings of the 5th ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP 2003)*. 241–252.
- Ilya Sergey and Dave Clarke. 2012. Gradual Ownership Types. In *Proceedings of the 21st European Symposium on Programming Languages and Systems (ESOP 2012) (Lecture Notes in Computer Science)*, Helmut Seidl (Ed.), Vol. 7211. Springer-Verlag, Tallinn, Estonia, 579–599.
- Jeremy Siek and Walid Taha. 2006. Gradual Typing for Functional Languages. In *Proceedings of the Scheme and Functional Programming Workshop*. 81–92.
- Jeremy Siek and Walid Taha. 2007. Gradual Typing for Objects. In *Proceedings of the 21st European Conference on Object-oriented Programming (ECOOP 2007) (Lecture Notes in Computer Science)*, Erik Ernst (Ed.). Springer-Verlag, Berlin, Germany, 2–27.
- Jeremy Siek and Philip Wadler. 2010. Threesomes, with and without blame. In *Proceedings of the 37th annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2010)*. ACM Press, Madrid, Spain, 365–376.
- Jeremy G. Siek, Michael M. Vitousek, Matteo Cimini, and John Tang Boyland. 2015a. Refined Criteria for Gradual Typing. In *1st Summit on Advances in Programming Languages (SNAPL 2015) (Leibniz International Proceedings in Informatics (LIPIcs))*, Vol. 32. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Asilomar, California, USA, 274–293.
- Jeremy G. Siek, Michael M. Vitousek, Matteo Cimini, Sam Tobin-Hochstadt, and Ronald Garcia. 2015b. Monotonic References for Efficient Gradual Typing. In *Proceedings of the 24th European Symposium on Programming Languages and Systems (ESOP 2015) (Lecture Notes in Computer Science)*, Jan Vitek (Ed.), Vol. 9032. Springer-Verlag, London, UK, 432–456.
- Jeremy G Siek and Philip Wadler. 2016. The key to blame : Gradual typing meets cryptography. Unpublished manuscript.
- Eijiro Sumii and Benjamin C. Pierce. 2004. A Bisimulation for Dynamic Sealing. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2005)*. ACM Press, Venice, Italy, 161–172.
- Asumu Takikawa, Daniel Feltey, Ben Greenman, Max S. New, Jan Vitek, and Matthias Felleisen. 2016. Is sound gradual typing dead?, See Bodík and Majumdar (2016), 456–468.
- Sam Tobin-Hochstadt and Matthias Felleisen. 2006. Interlanguage migration: from scripts to programs. In *Proceedings of the ACM Dynamic Languages Symposium (DLS 2006)*. ACM Press, Portland, Oregon, USA, 964–974.
- Matías Toro, Ronald Garcia, and Éric Tanter. 2018. Type-Driven Gradual Security with References. *ACM Transactions on Programming Languages and Systems* 40, 4 (Nov. 2018), 16:1–16:55.
- Matías Toro, Elizabeth Labrada, and Éric Tanter. 2019. Gradual Parametricity, Revisited. *Proceedings of the ACM on Programming Languages* 3, POPL (Jan. 2019), 17:1–17:30.

- Matías Toro and Éric Tanter. 2017. A Gradual Interpretation of Union Types. In *Proceedings of the 24th Static Analysis Symposium (SAS 2017) (Lecture Notes in Computer Science)*, Vol. 10422. Springer-Verlag, New York City, NY, USA, 382–404.
- Matías Toro and Éric Tanter. 2020. Abstracting Gradual References. *Science of Computer Programming* 197 (Oct. 2020), 1–65.
- Philip Wadler. 1989. Theorems for Free!. In *Proceedings of the Fourth International Conference on Functional Programming Languages and Computer Architecture (FPCA '89)*. ACM, London, United Kingdom, 347–359.
- Philip Wadler. 2017. Abstract Data Types without the Types. *Journal of Universal Computer Science* 23, 1 (2017), 5–20.
- Jenna Wise, Johannes Bader, Cameron Wong, Jonathan Aldrich, Éric Tanter, and Joshua Sunshine. 2020. Gradual Verification of Recursive Heap Data Structures. *Proceedings of the ACM on Programming Languages* 4, OOPSLA (Nov. 2020), 228:1–228:28.
- Roger Wolff, Ronald Garcia, Éric Tanter, and Jonathan Aldrich. 2011. Gradual Typestate. In *Proceedings of the 25th European Conference on Object-oriented Programming (ECOOP 2011) (Lecture Notes in Computer Science)*, Mira Mezini (Ed.), Vol. 6813. Springer-Verlag, Lancaster, UK, 459–483.
- Ningning Xie, Xuan Bi, and Bruno C. d. S. Oliveira. 2018. Consistent Subtyping for All. In *Proceedings of the 27th European Symposium on Programming Languages and Systems (ESOP 2018) (Lecture Notes in Computer Science)*, Amal Ahmed (Ed.), Vol. 10801. Springer-Verlag, Thessaloniki, Greece, 3–30.

Annexes

Annex A

Gradual Parametricity, Revisited

A.1 SF: Well-formedness

In this section we present auxiliary definitions for well-formedness of type name stores, and well-formedness of types.

Definition A.1 (Well-formedness of the type namn)

$$\frac{}{\vdash \cdot} \qquad \frac{\alpha \notin \Sigma \quad \Sigma; \cdot \vdash T \quad \vdash \Sigma}{\vdash \Sigma, \alpha : T}$$

Definition A.2 (Well-formedness on)

$$\frac{\vdash \Sigma}{\Sigma; \Delta \vdash B} \qquad \frac{\Sigma; \Delta \vdash T_1 \quad \Sigma; \Delta \vdash T_2}{\Sigma; \Delta \vdash T_1 \rightarrow T_2} \qquad \frac{\Sigma; \Delta, X \vdash T}{\Sigma; \Delta \vdash \forall X. T} \qquad \frac{\Sigma; \Delta \vdash T_1 \quad \Sigma; \Delta \vdash T_2}{\Sigma; \Delta \vdash T_1 \times T_2}$$
$$\frac{\vdash \Sigma \quad X \in \Delta}{\Sigma; \Delta \vdash X} \qquad \frac{\vdash \Sigma \quad \alpha : T \in \Sigma}{\Sigma; \Delta \vdash \alpha}$$

A.2 GSF: Statics

In this section we present auxiliary definitions and proofs of the statics semantics of GSF not presented in the paper.

A.2.1 Syntax and Syntactic Meaning of Gradual Types

Proposition A.3 (Type Precision, inductively) *The inductive definition of type precision given in Figure 7.3 is equivalent to Definition B.94.*

PROOF. Direct by induction on the type structure of G_1 and G_2 . We only present representative cases to illustrate the reasoning used in the proof. We prove first that $C(G_1) \subseteq C(G_2) \Rightarrow G_1 \sqsubseteq G_2$, where $G_1 \sqsubseteq G_2$ stands for the inductive definition given in Figure 3.4.

Case ($G_1 = B, G_2 = B$). Then $\{B\} \subseteq \{B\}$, but we already know that $B \sqsubseteq B$ and the result holds.

Case ($G_1 = G, G_2 = ?$). Then $C(G) \subseteq C(?) = \text{TYPE}$, but $G \sqsubseteq ?$ is an axiom and the result holds.

Case ($G_1 = \forall X.G'_1, G_2 = \forall X.G'_2$). Then we know that $\{\forall X.T \mid T \in C(G'_1)\} \subseteq \{\forall X.T \mid T \in C(G'_2)\}$, then it must be the case that $C(G'_1) \subseteq C(G'_2)$. Then by induction hypothesis $G_1 \sqsubseteq G_2$, then by inductive definition of precision for type abstractions, $\forall X.G_1 \sqsubseteq \forall X.G_2$ and the result holds.

Then we prove the other direction, i.e. $G_1 \sqsubseteq G_2 \Rightarrow C(G_1) \subseteq C(G_2)$.

Case ($G_1 = B, G_2 = B$). Then $B \sqsubseteq B$, but we already know that $\{B\} \subseteq \{B\}$ and the result holds.

Case ($G_1 = G, G_2 = ?$). Then $G \sqsubseteq ?$, but $C(G) \subseteq C(?) = \text{TYPE}$ and the result holds.

Case ($G_1 = \forall X.G'_1, G_2 = \forall X.G'_2$). Then we know that $\forall X.G_1 \sqsubseteq \forall X.G_2$, then by looking at the premise of the corresponding definition, $G'_1 \sqsubseteq G'_2$. Then by induction hypothesis $C(G'_1) \subseteq C(G'_2)$. But we have to prove that $\{\forall X.T \mid T \in C(G'_1)\} \subseteq \{\forall X.T \mid T \in C(G'_2)\}$, which is direct from $C(G'_1) \subseteq C(G'_2)$.

□

Proposition A.4 (Galois connection) *$\langle C, A \rangle$ is a Galois connection, i.e.:*

- a) (Soundness) *for any non-empty set of static types $S = \{\overline{T}\}$, we have $S \subseteq C(A(S))$*
- b) (Optimality) *for any gradual type G , we have $A(C(G)) \sqsubseteq G$.*

PROOF. We first proceed to prove a) by induction on the structure of the non-empty set S .

Case ($\{B\}$). Then $A(\{B\}) = B$. But $C(B) = \{B\}$ and the result holds.

Case ($\{\overline{T_{i1}} \rightarrow \overline{T_{i2}}\}$). Then $A(\{\overline{T_{i1}} \rightarrow \overline{T_{i2}}\}) = A(\{\overline{T_{i1}}\}) \rightarrow A(\{\overline{T_{i2}}\})$. But by definition of C , $C(A(\{\overline{T_{i1}}\}) \rightarrow A(\{\overline{T_{i2}}\})) = \{T_1 \rightarrow T_2 \mid T_1 \in C(A(\{\overline{T_{i1}}\})), T_2 \in C(A(\{\overline{T_{i2}}\}))\}$. By induction hypotheses, $\{\overline{T_{i1}}\} \subseteq C(A(\{\overline{T_{i1}}\}))$ and $\{\overline{T_{i2}}\} \subseteq C(A(\{\overline{T_{i2}}\}))$, therefore $\{\overline{T_{i1}} \rightarrow \overline{T_{i2}}\} \subseteq \{T_1 \rightarrow T_2 \mid T_1 \in \{\overline{T_{i1}}\}, T_2 \in \{\overline{T_{i2}}\}\} \subseteq \{T_1 \rightarrow T_2 \mid T_1 \in C(A(\{\overline{T_{i1}}\})), T_2 \in C(A(\{\overline{T_{i2}}\}))\}$ and the result holds.

Case ($\{\overline{T_{i1}} \times \overline{T_{i2}}\}$). We proceed analogous to case $\{\overline{T_{i1}} \rightarrow \overline{T_{i2}}\}$.

Case ($\{X\}, \{\alpha\}$). We proceed analogous to case $\{B\}$.

Case ($\{\overline{\forall X.T_i}\}$). Then $A(\{\overline{\forall X.T_i}\}) = \forall X.A(\{\overline{T_i}\})$. But by definition of C , $C(\forall X.A(\{\overline{T_i}\})) = \{\forall X.T \mid T \in C(A(\{\overline{T_i}\}))\}$. By induction hypothesis, $\{\overline{T_i}\} \subseteq C(A(\{\overline{T_i}\}))$, therefore $\{\overline{\forall X.T_i}\} = \{\forall X.T \mid T \in \{\overline{T_i}\}\} \subseteq \{\forall X.T \mid T \in C(A(\{\overline{T_i}\}))\}$ and the result holds.

Case ($\{\overline{T_i}\}$ heterogeneous). Then $A(\{\overline{T_i}\}) = ?$ and therefore $C(A(\{\overline{T_i}\})) = \text{TYPE}$, but $\{\overline{T_i}\} \subseteq \text{TYPE}$ and the result holds.

Now let us proceed to prove b) by induction on gradual type G .

Case (B). Trivial because $C(B) = \{B\}$, and $A(\{B\}) = B$.

Case ($G_1 \rightarrow G_2$). We have to prove that $A(C(G_1 \rightarrow G_2)) \sqsubseteq G_1 \rightarrow G_2$, which is equivalent to prove that $C(A(\widehat{T})) \subseteq \widehat{T}$, where $\widehat{T} = C(G_1 \rightarrow G_2) = \{T_1 \rightarrow T_2 \mid T_1 \in C(G_1), T_2 \in C(G_2)\}$. Then \widehat{T} has the form $\{\overline{T_{i1} \rightarrow T_{i2}}\}$, such that $\forall i, T_{i1} \in C(G_1)$ and $T_{i2} \in C(G_2)$. Also note that $\{\overline{T_{i1}}\} = C(G_1)$ and $\{\overline{T_{i2}}\} = C(G_2)$. But by definition of A , $A(\{\overline{T_{i1} \rightarrow T_{i2}}\}) = A(\{\overline{T_{i1}}\}) \rightarrow A(\{\overline{T_{i2}}\})$ and therefore $C(A(\{\overline{T_{i1} \rightarrow T_{i2}}\})) = \{T_1 \rightarrow T_2 \mid T_1 \in C(A(\{\overline{T_{i1}}\})), T_2 \in C(A(\{\overline{T_{i2}}\}))\}$. But by induction hypotheses $C(A(\{\overline{T_{i1}}\})) \subseteq C(G_1)$ and $C(A(\{\overline{T_{i2}}\})) \subseteq C(G_2)$ and the result holds.

Case ($G_1 \times G_2$). We proceed analogous to case $G_1 \rightarrow G_2$.

Case (X, α). We proceed analogous to case B .

Case ($\forall X.G$). We have to prove that $A(C(\forall X.G)) \sqsubseteq \forall X.G$, which is equivalent to prove that $C(A(\widehat{T})) \subseteq \widehat{T}$, where $\widehat{T} = C(\forall X.G) = \{\forall X.T \mid T \in C(G)\}$. Then \widehat{T} has the form $\{\overline{\forall X.T_i}\}$, such that $\forall i, T_i \in C(G)$. Also note that $\{\overline{T_i}\} = C(G)$. But by definition of A , $A(\{\overline{\forall X.T_i}\}) = \forall X.A(\{\overline{T_i}\})$ and therefore $C(A(\{\overline{\forall X.T_i}\})) = \{\forall X.T \mid T \in C(A(\{\overline{T_i}\}))\}$. But by induction hypothesis $C(A(\{\overline{T_i}\})) \subseteq C(G)$ and the result holds.

Case (?). Then we have to prove that $C(A(?)) \subseteq C(?) = \text{TYPE}$, but this is always true and the result holds immediately. □

A.2.2 Lifting the Static Semantics

Definition A.5 (Store prn) $\Xi_1 \sqsubseteq \Xi_2$ if and only if $\text{dom}(\Xi_1) = \text{dom}(\Xi_2)$ and $\forall \alpha \in \text{dom}(\Xi_1), \Xi_1(\alpha) \sqsubseteq \Xi_2(\alpha)$.

Lemma A.6 If $\Xi_1 \sqsubseteq \Xi_2, \vdash \Xi_i, G_1 \sqsubseteq G_2$, and $\Xi_1; \Delta \vdash G_1$, then $\Xi_2; \Delta \vdash G_2$.

PROOF. Straightforward induction on relation $G_1 \sqsubseteq G_2$. We only present interesting cases.

Case ($G_1 = \forall X.G'_1, G_2 = \forall X.G'_2$). By definition of precision $G'_1 \sqsubseteq G'_2$. By definition of well-formedness of types, $\Xi_1; X \vdash G'_1$ and then by induction hypothesis $\Xi_2; \Delta, X \vdash G'_2$. Then by definition of well-formedness of types $\Xi_2; \Delta \vdash \forall X.G'_2$ and the result holds.

Case ($G_2 = ?$). This is trivial because as $\vdash \Xi_2$, then $\Xi_2; \Delta \vdash ?$.

Case ($G_1 = \alpha, G_2 = \alpha$). Trivial by definition of $\Xi_1 \sqsubseteq \Xi_2, \alpha \in \text{dom}(\Xi_2)$, therefore $\alpha : G'_2 \in \Xi_2$ and then $\Xi_2; \Delta \vdash \alpha$. □

Lemma A.7 Let $\Xi_1 \sqsubseteq \Xi_2$, then $\vdash \Xi_1 \Rightarrow \vdash \Xi_2$.

PROOF. By induction on relation $\Xi_1 \sqsubseteq \Xi_2$.

Case $(\cdot \sqsubseteq \cdot)$. Trivial as $\vdash \cdot$.

Case $(\Xi'_1, \alpha : G_1 \sqsubseteq \Xi'_2, \alpha : G_2)$. By definition of store precision we know that $\Xi'_1 \sqsubseteq \Xi'_2$ and that $G_1 \sqsubseteq G_2$. By definition of well-formedness, $\vdash \Xi'_1, \alpha : G_1 \Rightarrow \vdash \Xi'_1$, therefore by induction hypothesis $\vdash \Xi'_2$. We only have left to prove is that $\Xi'_2; \cdot \vdash G_2$, which follows directly from Lemma A.6.

□

Lemma A.8 *If $\Sigma \in C(\Xi)$ and $\vdash \Sigma$, then $\vdash \Xi$*

PROOF. Corollary of Lemma A.7 as $\Sigma \sqsubseteq \Xi$.

□

Lemma A.9 *If $\Sigma; \Delta \vdash T_1 = T_2$, then $\Sigma; \Delta \vdash T_1$ and $\Sigma; \Delta \vdash T_2$.*

PROOF. By induction on relation $\Sigma; \Delta \vdash T_1 = T_2$. Most cases are straightforward, so we present only the interesting cases.

Case $(T_1 = \forall X.T'_1, T_2 = \forall X.T'_2)$. As $\Sigma; \Delta \vdash \forall X.T'_1 = \forall X.T'_2$, by inspection of the derivation rule, $\Sigma; \Delta, X \vdash T'_1 = T'_2$. By induction hypotheses we know that $\Sigma; \Delta, X \vdash T'_1$, and that $\Sigma; \Delta, X \vdash T'_2$. Therefore by well-formedness of types we know that $\Sigma; \Delta \vdash \forall X.T'_1$ and that $\Sigma; \Delta \vdash \forall X.T'_2$ and the result holds.

Case $(T_1 = X, T_2 = X)$. As $\Sigma; \Delta \vdash X = X$, then we know by inspection of the derivation rule that $\vdash \Sigma$ and that $X \in \Delta$. Then as $\vdash \Sigma$ and that $X \in \Delta$, $\Sigma; \Delta \vdash X$ and the result holds.

□

Proposition A.10 (Type Consistency, inductively) *The inductive definition of type consistency given in Figure 7.3 is equivalent to Definition 7.2.*

PROOF. First we prove that $\Sigma; \Delta \vdash T_1 = T_2$ for some $\Sigma \in C(\Xi)$, $T_i \in C(G_i)$ implies that $\Xi; \Delta \vdash G_1 \sim G_2$, where $\Xi; \Delta \vdash G_1 \sim G_2$ stands for the inductive definition of consistency. We proceed by straightforward induction on G_i such that the predicate holds (we only show interesting cases). By Lemma A.8 we know that if $\vdash \Sigma$ then $\vdash \Xi$, which will be assumed to be true whenever is needed.

Case $(G_1 = B, G_2 = B)$. Then $\Sigma; \Delta \vdash B = B$, but we already know that $\Xi \vdash B \sim B$ and the result holds.

Case $(G_1 = G, G_2 = ?)$. We know that $\Sigma; \Delta \vdash T_1 = T_2$ for some $T_1 \in C(G)$ and $T_2 \in C(?)$. Then by Lemma A.9, $\Sigma; \Delta \vdash T_1$, and as $\Sigma \sqsubseteq \Xi$ and $T_1 \sqsubseteq G$, by Lemma A.6, $\Xi; \Delta \vdash G$. Then as $\Xi; \Delta \vdash G$, $G \sim ? = \text{TYPE}$ and the result holds.

Case $(G_1 = \forall X.G'_1, G_2 = \forall X.G'_2)$. Then we know that $\Sigma; \Delta \vdash \forall X.T_1 = \forall X.T_2$ where $\forall X.T_1 \in C(\forall X.G'_1)$, $\forall X.T_2 \in C(\forall X.G'_2)$. Notice that $T_1 \in C(G'_1)$, $T_2 \in C(G'_2)$, and that $\Sigma; \Delta, X \vdash T_1 = T_2$. Then by induction hypotheses, $\Xi \vdash G'_1 \sim G'_2[\Delta, X]$, and therefore $\Xi; \Delta \vdash \forall X.G'_1 \sim \forall X.G'_2$ and the result holds.

Then we prove the other direction, i.e. $G_1 \sqsubseteq G_2 \Rightarrow C(G_1) \sim C(G_2)$.

Case $(G_1 = B, G_2 = B)$. Then $B \sqsubseteq B$, but we already know that $B \in C(B)$ and $\Sigma; \Delta \vdash B = B$, and the result holds immediately.

Case $(G_1 = G, G_2 = ?)$. Then $G \sqsubseteq ?$. Let $T_1 \in C(G)$ and $\Sigma \in C(\Xi)$ such that $\Sigma; \Delta \vdash T_1$. As $C(?) = \text{TYPE}$, we can choose $T_1 \in \text{TYPE}$, so $\Sigma; \Delta \vdash T_1 = T_1$, and the result holds.

Case $(G_1 = \forall X.G'_1, G_2 = \forall X.G'_2)$. Then we know that $\Xi; \Delta \vdash \forall X.G'_1 \sim \forall X.G'_2$, then by looking at the premise of the corresponding definition, $\Xi; \Delta, X \vdash G'_1 \sim G'_2$. Then by induction hypotheses $\exists T_1 \in C(G'_1), T_2 \in C(G'_2), \Sigma \in C(\Xi)$, such that $\Sigma; \Delta, X \vdash T_1 = T_2$. By definition of consistency $\forall X.T_i \in C(G_i)$. Then by definition of equality, $\Sigma; \Delta \vdash \forall X.T_1 = \forall X.T_2$ and the result holds. \square

Definition B.97 (Consistent lifting of functions) *Let F_n be a function of type $\text{TYPE}^n \rightarrow \text{TYPE}$. Its consistent lifting F_n^\sharp , of type $\text{GTYPE}^n \rightarrow \text{GTYPE}$, is defined as: $F_n^\sharp(\overline{G}) = A(\{F_n(\overline{T}) \mid \overline{T} \in \overline{C}(G)\})$.*

Lemma A.11 $G = A(C(G))$

PROOF. Then we have to prove that $G = A(C(G))$. By optimality (Prop 3.3.b), we know that $A(C(G)) \sqsubseteq G$, and by soundness (Prop 3.3.a), $C(G) \subseteq C(A(C(G)))$, i.e. $G \sqsubseteq A(C(G))$. Therefore $G \sqsubseteq A(C(G))$ and $A(C(G)) \sqsubseteq G$, thus $G = A(C(G))$ and the result holds. \square

Lemma A.12 $G[G'/X] = A(\{T[T'/X] \mid T \in C(G), T' \in C(G')\})$.

PROOF. We proceed by induction on G . We only present interesting cases.

Case $(G = X)$. Then $G[G'/X] = G'$, and $C(G) = \{X\}$. Then we have to prove that $G' = A(\{T' \mid T' \in C(G')\})$. But notice that $A(\{T' \mid T' \in C(G')\}) = A(C(G'))$ and by Lemma A.11 the result holds immediately.

Case $(G = ?)$. Then $G[G'/X] = ?$, and $C(G) = \text{TYPE}$. Then we have to prove that $? = A(\{T[T'/X] \mid T \in \text{TYPE}, T' \in C(G')\})$. But notice that $A(\{T[T'/X] \mid T \in \text{TYPE}, T' \in C(G')\}) = A(C(\text{TYPE}))$ and by Lemma A.11 the result holds immediately.

Case $(G = \forall Y.G'')$. Then $G[G'/X] = \forall Y.G''[G'/X]$, and $C(G) = \forall Y.C(G'')$. Then we have to prove that $\forall Y.G''[G'/X] = A(\{\forall Y.T''[T'/X] \mid T'' \in C(G''), T' \in C(G')\})$. But notice that by definition of abstraction $A(\{\forall Y.T''[T'/X] \mid T'' \in C(G''), T' \in C(G')\}) = \forall Y.A(\{T''[T'/X] \mid T'' \in C(G''), T' \in C(G')\})$. Then by induction hypothesis on G'' , $G''[G'/X] = A(\{T''[T'/X] \mid T'' \in C(G''), T' \in C(G')\})$, therefore $\forall Y.G''[G'/X] = \forall Y.A(\{T''[T'/X] \mid T'' \in C(G''), T' \in C(G')\})$ and the result holds. \square

Proposition A.13 (Consistent type functions) *The definitions of dom^\sharp , cod^\sharp , inst^\sharp , and proj_i^\sharp given in Fig. 3.4 are consistent liftings, as per Def. B.97, of the corresponding functions from Fig. 3.1.*

PROOF. We present the proof for inst^\sharp and dom^\sharp (the other proofs are analogous).

First we prove that $\text{inst}^\sharp(G, G') = A(\widehat{\text{inst}}(C^2(G, G')))$, where $\text{inst}^\sharp(G, G')$ correspond to the algorithmic definitions presented in Fig. 3.4. Notice that

$$\begin{aligned} & A(\widehat{\text{inst}}(C^2(G, G'))) \\ &= A(\widehat{\text{inst}}(\{\langle T, T' \rangle \mid T \in C(G), T' \in C(G')\})) \\ &= A(\{T[T'/X] \mid \forall X.T \in C(G), T' \in C(G')\}) \end{aligned}$$

But then the result follows immediately from Lemma A.12.

Then we prove that $dom^\sharp(G) = A(\widehat{dom}(C(G)))$, where $dom^\sharp(G)$ correspond to the algorithmic definitions presented in Fig. 3.4. We proceed by induction on G .

Case ($G = G_1 \rightarrow G_2$). Notice that

$$\begin{aligned}
& A(\widehat{dom}(C(G))) \\
&= A(\widehat{dom}(C(G_1 \rightarrow G_2))) \\
&= A(\widehat{dom}(\{T_1 \rightarrow T_2 \mid T_1 \in C(G_1), T_2 \in C(G_2)\})) \\
&= A(\{T_1 \mid T_1 \in C(G_1)\}) \\
&= A(C(G_1))
\end{aligned}$$

But $dom^\sharp(G_1 \rightarrow G_2) = G_1$. Then we have to prove that $G_1 = A(C(G_1))$ which holds immediately by Lemma A.11.

Case ($G = ?$). Notice that

$$\begin{aligned}
& A(\widehat{dom}(C(G))) \\
&= A(\widehat{dom}(C(?))) \\
&= A(\widehat{dom}(\text{TYPE})) \\
&= A(\text{TYPE}) \\
&= ?
\end{aligned}$$

and the result holds immediately as $dom^\sharp(?) = ?$.

Case ($G \neq ? \neq G_1 \rightarrow G_2$). If G has not the form $G_1 \rightarrow G_2$, or is not $?$, then $dom^\sharp(G)$ is undefined. Then as $\nexists T \in C(G)$ such that $T = T_1 \rightarrow T_2$ the result holds immediately as $dom(T)$ is undefined $\forall T \in C(G)$.

□

A.2.3 Well-formedness

In this section we present auxiliary definitions of the statics semantics of GSF.

Definition A.14 (Well-formedness of type namen)

$$\frac{}{\vdash \cdot} \qquad \frac{\alpha \notin \Xi \quad \Xi; \cdot \vdash G \quad \vdash \Xi}{\vdash \Xi, \alpha : G}$$

Definition A.15 (Well-formedness on)

$$\begin{array}{c}
\frac{\vdash \Xi}{\Xi; \Delta \vdash B} \qquad \frac{\Xi; \Delta \vdash G_1 \quad \Xi; \Delta \vdash G_2}{\Xi; \Delta \vdash G_1 \rightarrow G_2} \qquad \frac{\Xi; \Delta, X \vdash G}{\Xi; \Delta \vdash \forall X. G} \\
\\
\frac{\Xi; \Delta \vdash G_1 \quad \Xi; \Delta \vdash G_2}{\Xi; \Delta \vdash G_1 \times G_2} \qquad \frac{\vdash \Xi \quad X \in \Delta}{\Xi; \Delta \vdash X} \qquad \frac{\vdash \Xi \quad \alpha : G \in \Xi}{\Xi; \Delta \vdash \alpha} \qquad \frac{\vdash \Xi}{\Xi; \Delta \vdash ?}
\end{array}$$

A.2.4 Static Properties

In this section we present two static properties of GSF and the proof: the static equivalence for static terms and the static gradual guarantee.

Static Equivalence for Static Terms

Proposition A.16 (Static equivalence for static terms) *Let t be a static term and G a static type ($G = T$). We have $\vdash_S t : T$ if and only if $\vdash t : T$.*

PROOF. We prove this proposition for open terms instead. The proof is direct thanks to the equivalence between the typing rules and the equivalence between type equality and type consistency rules for static types. We only present one case to illustrate the reasoning.

First we prove $\Sigma; \Delta \vdash_S t : T \Rightarrow \Sigma; \Delta \vdash t : T$ by induction on judgment $\Sigma; \Delta \vdash_S t : T$.

Case $(\Sigma; \Delta \vdash_S t'[T''] : inst(\forall X.T', T''))$. Then $\Sigma; \Delta \vdash_S t' : \forall X.T'$, and by induction hypothesis $\Sigma; \Delta \vdash t' : \forall X.T'$. Then $inst^\sharp(\forall X.T', T'') = T[T''/X] = inst(\forall X.T', T'')$, and as $\Sigma; \Delta \vdash T''$, therefore $\Sigma; \Delta \vdash t'[T''] : T[T''/X]$ and the result holds.

Then we prove $\Sigma; \Delta \vdash t : T \Rightarrow \Sigma; \Delta \vdash_S t : T$ by induction on judgment $\Sigma; \Delta \vdash t : T$.

Case $(\Sigma; \Delta \vdash t'[T''] : inst^\sharp(\forall X.T', T''))$. Then $\Sigma; \Delta \vdash t' : \forall X.T'$, and by induction hypothesis $\Sigma; \Delta \vdash_S t' : \forall X.T'$. Then $inst(\forall X.T', T'') = T[T''/X] = inst^\sharp(\forall X.T', T'')$, and as $\Sigma; \Delta \vdash T''$, therefore $\Sigma; \Delta \vdash_S t'[T''] : T[T''/X]$ and the result holds.

□

Static Gradual Guarantee

In this section we present the proof of the static gradual guarantee property. In the Definition A.17 and Definition A.18 we present term precision and type environment precision.

Definition A.17 (Term precision)

$$\begin{array}{c}
 (Px) \frac{}{x \sqsubseteq x} \qquad (Pb) \frac{}{b \sqsubseteq b} \qquad (P\lambda) \frac{t \sqsubseteq t' \quad G \sqsubseteq G'}{(\lambda x : G.t) \sqsubseteq (\lambda x : G'.t')} \\
 (P\Lambda) \frac{t \sqsubseteq t'}{(\Lambda X.t) \sqsubseteq (\Lambda X.t')} \qquad (Ppair) \frac{t_1 \sqsubseteq t'_1 \quad t_2 \sqsubseteq t'_2}{\langle t_1, t_2 \rangle \sqsubseteq \langle t'_1, t'_2 \rangle} \qquad (Pasc) \frac{t \sqsubseteq t' \quad G \sqsubseteq G'}{(t :: G) \sqsubseteq (t' :: G')} \\
 (Pop) \frac{\bar{t} \sqsubseteq \bar{t}'}{op(\bar{t}) \sqsubseteq op(\bar{t}')} \qquad (Papp) \frac{t_1 \sqsubseteq t'_1 \quad t_2 \sqsubseteq t'_2}{t_1 t_2 \sqsubseteq t'_1 t'_2} \qquad (PappG) \frac{t \sqsubseteq t' \quad G \sqsubseteq G'}{t [G] \sqsubseteq t' [G']} \\
 (Ppairi) \frac{t \sqsubseteq t'}{\pi_i(t) \sqsubseteq \pi_i(t')}
 \end{array}$$

Definition A.18 (Type environment precision)

$$\frac{}{\cdot \sqsubseteq \cdot} \qquad \frac{\Gamma \sqsubseteq \Gamma' \quad G \sqsubseteq G'}{\Gamma, x : G \sqsubseteq \Gamma', x : G'}$$

Lemma A.19 *If $\Xi; \Delta; \Gamma \vdash t : G$ and $\Gamma \sqsubseteq \Gamma'$, then $\Xi; \Delta; \Gamma' \vdash t : G'$ for some $G \sqsubseteq G'$.*

PROOF. Simple induction on type derivation $\Xi; \Delta; \Gamma \vdash t : G$ (we only present interesting cases).

Case ($t = x$). we know that $\Sigma; \Delta; \Gamma \vdash x : G$ and $\Gamma(x) = G$. By definition of $\Gamma \sqsubseteq \Gamma'$, $\Gamma(x) \sqsubseteq \Gamma'(x)$, therefore $\Sigma; \Delta; \Gamma \vdash x : G'$, where $G \sqsubseteq G'$ and the result holds.

Case ($t = (\lambda x : G_1.t')$). we know that $\Sigma; \Delta; \Gamma \vdash (\lambda x : G_1.t') : G_1 \rightarrow G_2$, where $\Sigma; \Delta; \Gamma, x : G_1 \vdash t' : G_2$. As $\Gamma \sqsubseteq \Gamma'$ and $G_1 \sqsubseteq G_1$, then by definition of precision for type environments, $\Gamma, x : G_1 \sqsubseteq \Gamma', x : G_1'$. Therefore by induction hypothesis on $\Sigma; \Delta; \Gamma, x : G_1 \vdash t' : G_2$, $\Sigma; \Delta; \Gamma', x : G_1 \vdash t' : G_2'$, where $G_2 \sqsubseteq G_2'$. Finally, by (G λ), $\Sigma; \Delta; \Gamma' \vdash (\lambda x : G_1.t') : G_1 \rightarrow G_2'$, and as $G_1 \rightarrow G_2 \sqsubseteq G_1 \rightarrow G_2'$, the result holds. □

Lemma A.20 *If $\Xi; \Delta \vdash G_1 \sim G_2$ and $G_1 \sqsubseteq G_1'$ and $G_2 \sqsubseteq G_2'$ then $\Xi; \Delta \vdash G_1' \sim G_2'$.*

PROOF. By definition of $\Xi; \Delta \vdash \cdot \sim \cdot$, there exists $\langle T_1, T_2 \rangle \in C^2(G_1, G_2)$ such that $T_1 = T_2$. $G_1 \sqsubseteq G_1'$ and $G_2 \sqsubseteq G_2'$ mean that $C(G_1) \subseteq C(G_1')$ and $C(G_2) \subseteq C(G_2')$, therefore $\langle T_1, T_2 \rangle \in C^2(G_1', G_2')$, and the result follows. □

Lemma A.21 *If $G_1 \sqsubseteq G_1'$ and $G_2 \sqsubseteq G_2'$ then $G_1[G_2/X] \sqsubseteq G_1'[G_2'/X]$.*

PROOF. By induction on the relation of $G_1 \sqsubseteq G_1'$. We only present interesting cases.

Case ($X \sqsubseteq X$). Then we have to prove that $X[G_2/X] \sqsubseteq X[G_2'/X]$, which is equivalent to $G_2 \sqsubseteq G_2'$, but that is part of the premise and the result holds immediately.

Case ($G_1 \sqsubseteq ?$). Then we have to prove that $G_1[G_2/X] \sqsubseteq ?$ which is always true.

Case ($\forall Y.G_3 \sqsubseteq \forall Y.G_3'$). Then we have to prove that $\forall Y.G_3[G_2/X] \sqsubseteq \forall Y.G_3'[G_2'/X]$, which is equivalent to prove that $G_3[G_2/X] \sqsubseteq G_3'[G_2'/X]$, which holds by induction hypothesis on $G_3 \sqsubseteq G_3'$. □

Lemma A.22 *If $G_1 \sqsubseteq G_1'$ and $G_2 \sqsubseteq G_2'$ then $inst^\sharp(G_1, G_2) \sqsubseteq inst^\sharp(G_1', G_2')$.*

PROOF. By induction on relation $G_1 \sqsubseteq G_1'$.

Case ($? \sqsubseteq ?$). The result is trivial as $inst^\sharp(?, G_i') = ?$ and $? \sqsubseteq ?$.

Case ($\forall X.G_1 \sqsubseteq ?, \forall X.G_1 \sqsubseteq \forall X.G_2$). The result follows directly from Lemma A.21. □

Lemma A.23 *If $G_1 \sqsubseteq G_2$ then $proj_i^\sharp(G_1) \sqsubseteq proj_i^\sharp(G_2)$.*

PROOF. The proof is direct, analogous to Lemma A.22, by induction on relation $G_1 \sqsubseteq G_2$. □

Proposition A.24 (Static gradual guarantee for open terms) *If $\Xi; \Delta; \Gamma \vdash t_1 : G_1$ and $t_1 \sqsubseteq t_2$, then $\Xi; \Delta; \Gamma \vdash t_2 : G_2$, for some G_2 such that $G_1 \sqsubseteq G_2$.*

PROOF. We prove the property on opens terms instead of closed terms: If $\Xi; \Delta; \Gamma \vdash t_1 : G_1$ and $t_1 \sqsubseteq t_2$ then $\Xi; \Delta; \Gamma \vdash t_2 : G_2$ and $G_1 \sqsubseteq G_2$.

The proof proceed by induction on the typing derivation.

Case (Gx, Gb). Trivial by definition of term precision (\sqsubseteq) using (Px) , (Pb) respectively.

Case (Gλ). Then $t_1 = (\lambda x : G'_1.t)$ and $G_1 = G'_1 \widehat{\rhd} G'_2$. By $(G\lambda)$ we know that:

$$(G\lambda) \frac{\Xi; \Delta; \Gamma, x : G'_1 \vdash t : G'_2}{\Xi; \Delta; \Gamma \vdash \lambda x : G'_1.t : G'_1 \rightarrow G'_2} \quad (\text{A.1})$$

Consider t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = (\lambda x : G''_1.t')$ and therefore

$$(P\lambda) \frac{t \sqsubseteq t' \quad G'_1 \sqsubseteq G''_1}{(\lambda x : G'_1.t) \sqsubseteq (\lambda x : G''_1.t')} \quad (\text{A.2})$$

Using induction hypotheses on the premises of (A.1) and (B.2), $\Xi; \Delta; \Gamma, x : G'_1 \vdash t' : G''_2$ with $G'_2 \sqsubseteq G''_2$. By Lemma A.19, $\Xi; \Delta; \Gamma, x : G'_1 \vdash t' : G'''_2$ where $G''_2 \sqsubseteq G'''_2$. Then we can use rule $(G\lambda)$ to derive:

$$(G\lambda) \frac{\Xi; \Delta; \Gamma, x : G'_1 \vdash t' : G'''_2}{\Xi; \Delta; \Gamma \vdash (\lambda x : G'_1.t') : G'_1 \widehat{\rhd} G'''_2}$$

Where $G_2 \sqsubseteq G'''_2$. Using the premise of (B.2) and the definition of type precision we can infer that

$$G'_1 \widehat{\rhd} G'_2 \sqsubseteq G'_1 \widehat{\rhd} G'''_2$$

and the result holds.

Case (GΛ). Then $t_1 = (\Lambda X.t)$ and $G_1 = \forall X.G'_1$. By $(G\Lambda)$ we know that:

$$(G\Lambda) \frac{\Xi; \Delta, X; \Gamma \vdash t : G'_1}{\Xi; \Delta; \Gamma \vdash \Lambda X.t : \forall X.G'_1} \quad (\text{A.3})$$

Consider t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = (\Lambda X.t')$ and therefore

$$(P\Lambda) \frac{t \sqsubseteq t'}{(\Lambda X.t) \sqsubseteq (\Lambda X.t')} \quad (\text{A.4})$$

Using induction hypotheses on the premises of (B.3) and (B.4), $\Xi; \Delta, X; \Gamma \vdash t' : G''_1$ with $G'_1 \sqsubseteq G''_1$. Then we can use rule $(G\Lambda)$ to derive:

$$(G\Lambda) \frac{\Xi; \Delta, X; \Gamma \vdash t' : G''_1}{\Xi; \Delta; \Gamma \vdash (\Lambda X.t') : \forall X.G''_1}$$

Using the definition of type precision we can infer that

$$\forall X.G'_1 \sqsubseteq \forall X.G''_1$$

and the result holds.

Case (Gpair). Then $t_1 = \langle t'_1, t'_2 \rangle$ and $G_1 = G'_1 \times G'_2$. By (Gpair) we know that:

$$(Gpair) \frac{\Xi; \Delta; \Gamma \vdash t'_1 : G'_1 \quad \Xi; \Delta; \Gamma \vdash t'_2 : G'_2}{\Xi; \Delta; \Gamma \vdash t'_1 t'_2 : G'_1 \times G'_2} \quad (A.5)$$

Consider t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision, t_2 must have the form $\langle t''_1, t''_2 \rangle$ and therefore

$$(Ppair) \frac{t'_1 \sqsubseteq t''_1 \quad t'_2 \sqsubseteq t''_2}{\langle t'_1, t'_2 \rangle \sqsubseteq \langle t''_1, t''_2 \rangle} \quad (A.6)$$

Using induction hypotheses on the premises of (B.5) and (B.6), $\Xi; \Delta; \Gamma \vdash t''_1 : G''_1$ and $\Xi; \Delta; \Gamma \vdash t''_2 : G''_2$, where $G'_1 \sqsubseteq G''_1$ and $G'_2 \sqsubseteq G''_2$. Then we can use rule (Gpair) to derive:

$$(Gpair) \frac{\Xi; \Delta; \Gamma \vdash t''_1 : G''_1 \quad \Xi; \Delta; \Gamma \vdash t''_2 : G''_2}{\Xi; \Delta; \Gamma \vdash \langle t''_1, t''_2 \rangle : G''_1 \times G''_2}$$

Finally, using the definition of type precision we can infer that

$$G'_1 \times G'_2 \sqsubseteq G''_1 \times G''_2$$

and the result holds.

Case (Gasc). Then $t_1 = t :: G_1$. By (Gasc) we know that:

$$(Gasc) \frac{\Xi; \Delta; \Gamma \vdash t : G \quad \Xi; \Delta \vdash G \sim G_1}{\Xi; \Delta; \Gamma \vdash t :: G_1 : G_1} \quad (A.7)$$

Consider t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = t' :: G_2$ and therefore

$$(Pasc) \frac{t \sqsubseteq t' \quad G_1 \sqsubseteq G_2}{t :: G_1 \sqsubseteq t' :: G_2} \quad (A.8)$$

Using induction hypotheses on the premises of (B.7) and (B.8), $\Xi; \Delta; \Gamma \vdash t' : G'$ where $G \sqsubseteq G'$. We can use rule (Gasc) and Lemma A.20 to derive:

$$(Gasc) \frac{\Xi; \Delta; \Gamma \vdash t' : G' \quad \Xi; \Delta \vdash G' \sim G_2}{\Xi; \Delta; \Gamma \vdash t' :: G_2 : G_2}$$

Where $G_1 \sqsubseteq G_2$ and the result holds.

Case (Cop). Then $t_1 = op(\bar{t})$ and $G_1 = G^*$. By (Gop) we know that:

$$(Gop) \frac{\Xi; \Delta; \Gamma \vdash \bar{t} : \bar{G} \quad ty(op) = \bar{G}_2 \rightarrow G^*}{\Xi; \Delta \vdash \bar{G} \sim \bar{G}_2} \quad (A.9)$$

$$\frac{\Xi; \Delta; \Gamma \vdash \bar{t} : \bar{G} \quad ty(op) = \bar{G}_2 \rightarrow G^*}{\Xi; \Delta; \Gamma \vdash op(\bar{t}) : G^*}$$

Consider t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = op(\bar{t}')$ and therefore

$$(Pop) \frac{\bar{t} \sqsubseteq \bar{t}'}{op(\bar{t}) \sqsubseteq op(\bar{t}')} \quad (A.10)$$

Using induction hypotheses on the premises of (A.9) and (A.10), $\Xi; \Delta; \Gamma \vdash \bar{t}' : \bar{G}'$, where $\bar{G} \sqsubseteq \bar{G}'$. Using the Lemma A.20 we know that $\Xi; \Delta \vdash \bar{G}' \sim \bar{G}_2$. Therefore we can use rule (Gop) to derive:

$$(Gop) \frac{\Xi; \Delta; \Gamma \vdash \bar{t}' : \bar{G}' \quad ty(op) = \bar{G}_2 \rightarrow G^*}{\Xi; \Delta \vdash \bar{G}' \sim \bar{G}_2} \quad (A.9)$$

$$\frac{\Xi; \Delta; \Gamma \vdash \bar{t}' : \bar{G}' \quad ty(op) = \bar{G}_2 \rightarrow G^*}{\Xi; \Delta; \Gamma \vdash op(\bar{t}') : G^*}$$

and the result holds.

Case (Gapp). Then $t_1 = t'_1 t'_2$ and $G_1 = \text{cod}^\sharp(G'_1)$. By (Gapp) we know that:

$$\text{(Gapp)} \frac{\begin{array}{c} \Xi; \Delta; \Gamma \vdash t'_1 : G'_1 \quad \Xi; \Delta; \Gamma \vdash t'_2 : G'_2 \\ \Xi; \Delta \vdash \text{dom}^\sharp(G'_1) \sim G'_2 \end{array}}{\Xi; \Delta; \Gamma \vdash t'_1 t'_2 : \text{cod}^\sharp(G'_1)} \quad (\text{A.11})$$

Consider t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = t''_1 t''_2$ and therefore

$$\text{(Papp)} \frac{t'_1 \sqsubseteq t''_1 \quad t'_2 \sqsubseteq t''_2}{t'_1 t'_2 \sqsubseteq t''_1 t''_2} \quad (\text{A.12})$$

Using induction hypotheses on the premises of (B.9) and (B.10), $\Xi; \Delta; \Gamma \vdash t''_1 : G''_1$ and $\Xi; \Delta; \Gamma \vdash t''_2 : G''_2$, where $G'_1 \sqsubseteq G''_1$ and $G'_2 \sqsubseteq G''_2$. By definition type precision and the definition of dom^\sharp , $\text{dom}^\sharp(G'_1) \sqsubseteq \text{dom}^\sharp(G''_1)$ and, therefore by Lemma A.20, $\Xi; \Delta \vdash \text{dom}^\sharp(G'_1) \sim G'_2$. Also, by the previous argument $\text{cod}^\sharp(G'_1) \sqsubseteq \text{cod}^\sharp(G''_1)$. Then we can use rule (Gapp) to derive:

$$\text{(Gapp)} \frac{\begin{array}{c} \Xi; \Delta; \Gamma \vdash t''_1 : G''_1 \quad \Xi; \Delta; \Gamma \vdash t''_2 : G''_2 \\ \Xi; \Delta \vdash \text{dom}^\sharp(G''_1) \sim G''_2 \end{array}}{\Xi; \Delta; \Gamma \vdash t''_1 t''_2 : \text{cod}^\sharp(G''_1)}$$

and the result holds.

Case (GappG). Then $t_1 = t [G]$. By (GappG) we know that:

$$\text{(GappG)} \frac{\begin{array}{c} \Xi; \Delta; \Gamma \vdash t : G'_1 \quad \Xi; \Delta \vdash G \end{array}}{\Xi; \Delta; \Gamma \vdash t [G] : \text{inst}^\sharp(G'_1, G)} \quad (\text{A.13})$$

where $G_1 = \text{inst}^\sharp(G'_1, G)$. Consider t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = t' [G']$ and therefore

$$\text{(PappG)} \frac{t \sqsubseteq t' \quad G \sqsubseteq G'}{t [G] \sqsubseteq t' [G']} \quad (\text{A.14})$$

Using induction hypotheses on the premises of (B.11) and (B.12), $\Xi; \Delta; \Gamma \vdash t' : G'_2$ where $G'_1 \sqsubseteq G'_2$. We can use rule (GappG) and Lemma A.6 to derive:

$$\text{(Gasc)} \frac{\begin{array}{c} \Xi; \Delta; \Gamma \vdash t' : G'_2 \quad \Xi; \Delta \vdash G' \end{array}}{\Xi; \Delta; \Gamma \vdash t' [G'] : \text{inst}^\sharp(G'_2, G')}$$

Finally, by the Lemma A.22 we know that $\text{inst}^\sharp(G'_1, G) \sqsubseteq \text{inst}^\sharp(G'_2, G')$ and the result holds.

Case (Gpairi). Then $t_1 = \pi_i(t)$ and $G_1 = \text{proj}_i^\sharp(G)$. By (Gpair) we know that:

$$\text{(Gpairi)} \frac{\Xi; \Delta; \Gamma \vdash t : G}{\Xi; \Delta; \Gamma \vdash \pi_i(t) : \text{proj}_i^\sharp(G)} \quad (\text{A.15})$$

Consider t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision, t_2 must have the form $\pi_i(t')$ and therefore

$$\text{(Ppairi)} \frac{t \sqsubseteq t'}{\pi_i(t) \sqsubseteq \pi_i(t')} \quad (\text{A.16})$$

Using induction hypotheses on the premises of (B.13) and (B.14), $\Xi; \Delta; \Gamma \vdash t' : G'$ where $G \sqsubseteq G'$. Then we can use rule (Gpairi) to derive:

$$\text{(Gpairi)} \frac{\Xi; \Delta; \Gamma \vdash t' : G'}{\Xi; \Delta; \Gamma \vdash \pi_i(t') : \text{proj}_i^\sharp(G')}$$

Finally, by the Lemma A.23 we can infer that $\text{proj}_i^\sharp(G) \sqsubseteq \text{proj}_i^\sharp(G')$ and the result holds.

□

Proposition A.25 (Static gradual guarantee) *Let t and t' be closed GSF terms such that $t \sqsubseteq t'$ and $\vdash t : G$. Then $\vdash t' : G'$ and $G \sqsubseteq G'$.*

PROOF. Direct corollary of Prop. A.24.

□

A.3 GSF: Dynamics

In this section, we expose auxiliary definitions of the dynamic semantics of GSF. First, we present type precision, interior and consistent transitivity definitions for evidence types. Then we show some important definitions, used in the dynamic semantics of $\text{GSF}\varepsilon$. Finally, we present the translation semantics from GSF to $\text{GSF}\varepsilon$.

A.3.1 Evidence Type Precision

Figure A.1 presents the definition of the evidence type precision.

$E \sqsubseteq E$ Type precision			
$\frac{}{B \sqsubseteq B}$	$\frac{}{X \sqsubseteq X}$	$\frac{E_1 \sqsubseteq E'_1 \quad E_2 \sqsubseteq E'_2}{E_1 \rightarrow E_2 \sqsubseteq E'_1 \rightarrow E'_2}$	$\frac{E_1 \sqsubseteq E_2}{\forall X.E_1 \sqsubseteq \forall X.E_2}$
$\frac{E_1 \sqsubseteq E'_1 \quad E_2 \sqsubseteq E'_2}{E_1 \times E_2 \sqsubseteq E'_1 \times E'_2}$		$\frac{E_1 \sqsubseteq E_2}{\alpha^{E_1} \sqsubseteq \alpha^{E_2}}$	$\frac{}{E \sqsubseteq ?}$

Figure A.1: Evidence Type Precision

A.3.2 Initial Evidence

In Figure A.2 we present the interior function, used to compute the initial evidence.

A.3.3 Consistent Transitivity

In Figure A.3, we present the definition of consistent transitivity for evidence types.

A.3.4 $\text{GSF}\varepsilon$: Dynamic Semantics

In this section, we show the function definitions used in the dynamic semantics of $\text{GSF}\varepsilon$, specifically in the type application rule (RappG).

Definition A.26

$$\varepsilon_{out} \triangleq \langle E_*[\alpha^E], E_*[E'] \rangle \quad \text{where } E_* = \text{lift}_{\Xi}(\text{unlift}(\pi_2(\varepsilon))), \alpha^E = \text{lift}_{\Xi'}(\alpha), E' = \text{lift}_{\Xi}(G')$$

Definition A.27 $\langle E_1, E_2 \rangle [E_3] = \langle E_1[E_3], E_2[E_3] \rangle$

$$\boxed{\mathcal{G}(\cdot, \cdot) \text{ETYPE} \times \text{ETYPE} \rightarrow \text{EVIDENCE}}$$

$$\begin{array}{c}
\frac{E \in \text{BASETYPE} \cup \text{TYPEVAR} \cup \{?\}}{\mathcal{G}(\cdot, ()E, E) = \mathcal{G}(\cdot, ()?, E) = \mathcal{G}(\cdot, ()E, ?) = \langle E, E \rangle} \qquad \frac{\mathcal{G}(\cdot, ()E_1, E_2) = \langle E'_1, E'_2 \rangle}{\mathcal{G}(\cdot, ()\alpha^{E_1}, E_2) = \langle \alpha^{E'_1}, E'_2 \rangle} \\
\\
\frac{\mathcal{G}(\cdot, ()E_1, E_2) = \langle E'_1, E'_2 \rangle}{\mathcal{G}(\cdot, ()E_1, \alpha^{E_2}) = \langle E'_1, \alpha^{E'_2} \rangle} \qquad \frac{\mathcal{G}(\cdot, ()E_{11} \rightarrow E_{12}, ? \rightarrow ?) = \langle E'_1, E'_2 \rangle}{\mathcal{G}(\cdot, ()E_{11} \rightarrow E_{12}, ?) = \langle E'_1, E'_2 \rangle} \\
\\
\frac{\mathcal{G}(\cdot, ()? \rightarrow ?, E_{11} \rightarrow E_{12}) = \langle E'_1, E'_2 \rangle}{\mathcal{G}(\cdot, ()?, E_{11} \rightarrow E_{12}) = \langle E'_1, E'_2 \rangle} \qquad \frac{\mathcal{G}(\cdot, ()\forall X.E, \forall X.?) = \langle E'_1, E'_2 \rangle}{\mathcal{G}(\cdot, ()\forall X.E, ?) = \langle E'_1, E'_2 \rangle} \\
\\
\frac{\mathcal{G}(\cdot, ()\forall X.?, \forall X.E) = \langle E'_1, E'_2 \rangle}{\mathcal{G}(\cdot, ()?, \forall X.E) = \langle E'_1, E'_2 \rangle} \qquad \frac{\mathcal{G}(\cdot, ()E_{11} \times E_{12}, ? \times ?) = \langle E'_{11} \times E'_{12}, E'_{21} \times E'_{22} \rangle}{\mathcal{G}(\cdot, ()E_{11} \times E_{12}, ?) = \langle E'_{11} \times E'_{12}, E'_{21} \times E'_{22} \rangle} \\
\\
\frac{\mathcal{G}(\cdot, ()? \times ?, E_{11} \times E_{12}) = \langle E'_{11} \times E'_{12}, E'_{21} \times E'_{22} \rangle}{\mathcal{G}(\cdot, ()?, E_{11} \times E_{12}) = \langle E'_{11} \times E'_{12}, E'_{21} \times E'_{22} \rangle} \\
\\
\frac{\mathcal{G}(\cdot, ()E_{21}, E_{11}) = \langle E'_{21}, E'_{11} \rangle \quad \mathcal{G}(\cdot, ()E_{12}, E_{22}) = \langle E'_{12}, E'_{22} \rangle}{\mathcal{G}(\cdot, ()E_{11} \rightarrow E_{12}, E_{21} \rightarrow E_{22}) = \langle E'_{11} \rightarrow E'_{12}, E'_{21} \rightarrow E'_{22} \rangle} \\
\\
\frac{\mathcal{G}(\cdot, ()E_{11}, E_{21}) = \langle E'_{11}, E'_{21} \rangle \quad \mathcal{G}(\cdot, ()E_{12}, E_{22}) = \langle E'_{12}, E'_{22} \rangle}{\mathcal{G}(\cdot, ()E_{11} \times E_{12}, E_{21} \times E_{22}) = \langle E'_{11} \times E'_{12}, E'_{21} \times E'_{22} \rangle} \\
\\
\frac{\mathcal{G}(\cdot, ()E_1, E_2) = \langle E'_1, E'_2 \rangle}{\mathcal{G}(\cdot, ()\forall X.E_1, \forall X.E_2) = \langle \forall X.E'_1, \forall X.E'_2 \rangle}
\end{array}$$

Figure A.2: GSF: Computing Initial Evidence

Definition A.28

$$s[\alpha^E/X] = \begin{cases} b & s = b \\ \lambda x : G_1[\alpha/X].t[\alpha^E/X] & s = \lambda x : G_1.t \\ \Lambda Y.t[\alpha^E/X] & s = \Lambda Y.t \\ \langle s_1[\alpha^E/X], s_2[\alpha^E/X] \rangle & s = \langle s_1, s_2 \rangle \\ x & s = x \\ \varepsilon[\alpha^E/X]t[\alpha^E/X] :: G[\alpha/X] & s = \varepsilon t :: G \\ \overline{op(t[\alpha^E/X])} & s = op(\overline{t}) \\ t_1[\alpha^E/X] t_2[\alpha^E/X] & s = t_1 t_2 \\ \pi_i(t[\alpha^E/X]) & s = \pi_i(t) \\ t[\alpha^E/X] [G[\alpha/X]] & s = t [G] \end{cases}$$

$$\begin{array}{c}
\text{(base)} \frac{}{\langle B, B \rangle \mathbin{;} \langle B, B \rangle = \langle B, B \rangle} \qquad \text{(typeVar)} \frac{}{\langle X, X \rangle \mathbin{;} \langle X, X \rangle = \langle X, X \rangle} \\
\text{(idL)} \frac{}{\langle E_1, E_2 \rangle \mathbin{;} \langle ?, ? \rangle = \langle E_1, E_2 \rangle} \qquad \text{(idR)} \frac{}{\langle ?, ? \rangle \mathbin{;} \langle E_1, E_2 \rangle = \langle E_1, E_2 \rangle} \\
\text{(sealL)} \frac{\langle E_1, E_2 \rangle \mathbin{;} \langle E_3, E_4 \rangle = \langle E'_1, E'_2 \rangle}{\langle E_1, E_2 \rangle \mathbin{;} \langle E_3, \alpha^{E_4} \rangle = \langle E'_1, \alpha^{E'_2} \rangle} \qquad \text{(sealR)} \frac{\langle E_1, E_2 \rangle \mathbin{;} \langle E_3, E_4 \rangle = \langle E'_1, E'_2 \rangle}{\langle \alpha^{E_1}, E_2 \rangle \mathbin{;} \langle E_3, E_4 \rangle = \langle \alpha^{E'_1}, E'_2 \rangle} \\
\text{(unsl)} \frac{\langle E_1, E_2 \rangle \mathbin{;} \langle E_3, E_4 \rangle = \langle E'_1, E'_2 \rangle}{\langle E_1, \alpha^{E_2} \rangle \mathbin{;} \langle \alpha^{E_3}, E_4 \rangle = \langle E'_1, E'_2 \rangle} \\
\text{(func)} \frac{\langle E_{41}, E_{31} \rangle \mathbin{;} \langle E_{21}, E_{11} \rangle = \langle E_3, E_1 \rangle \quad \langle E_{12}, E_{22} \rangle \mathbin{;} \langle E_{32}, E_{42} \rangle = \langle E_2, E_4 \rangle}{\langle E_{11} \rightarrow E_{12}, E_{21} \rightarrow E_{22} \rangle \mathbin{;} \langle E_{31} \rightarrow E_{32}, E_{41} \rightarrow E_{42} \rangle = \langle E_1 \rightarrow E_2, E_3 \rightarrow E_4 \rangle} \\
\text{(abst)} \frac{\langle E_1, E_2 \rangle \mathbin{;} \langle E_3, E_4 \rangle = \langle E'_1, E'_2 \rangle}{\langle \forall X. E_1, \forall X. E_2 \rangle \mathbin{;} \langle \forall X. E_3, \forall X. E_4 \rangle = \langle \forall X. E'_1, \forall X. E'_2 \rangle} \\
\text{(pair)} \frac{\langle E_{11}, E_{21} \rangle \mathbin{;} \langle E_{31}, E_{41} \rangle = \langle E_1, E_3 \rangle \quad \langle E_{12}, E_{22} \rangle \mathbin{;} \langle E_{32}, E_{42} \rangle = \langle E_2, E_4 \rangle}{\langle E_{11} \times E_{12}, E_{21} \times E_{22} \rangle \mathbin{;} \langle E_{31} \times E_{32}, E_{41} \times E_{42} \rangle = \langle E_1 \times E_2, E_3 \times E_4 \rangle}
\end{array}$$

Figure A.3: GSF: Consistent Transitivity

Definition A.29

$$\text{lift}_{\Xi}(G) = \begin{cases} \text{lift}_{\Xi}(G_1) \rightarrow \text{lift}_{\Xi}(G_2) & G = G_1 \rightarrow G_2 \\ \forall X. \text{lift}_{\Xi}(G_1) & G = \forall X. G_1 \\ \text{lift}_{\Xi}(G_1) \times \text{lift}_{\Xi}(G_2) & G = G_1 \times G_2 \\ \alpha^{\text{lift}_{\Xi}(\Xi(\alpha))} & G = \alpha \\ G & \text{otherwise} \end{cases}$$

Definition A.30

$$\text{unlift}(E) = \begin{cases} B & E = B \\ \text{unlift}(E_1) \rightarrow \text{unlift}(E_2) & E = E_1 \rightarrow E_2 \\ \forall X. \text{unlift}(E_1) & E = \forall X. E_1 \\ \text{unlift}(E_1) \times \text{unlift}(E_2) & E = E_1 \times E_2 \\ \alpha & E = \alpha^{E_1} \\ X & E = X \\ ? & E = ? \end{cases}$$

A.3.5 Translation from GSF to GSF ε

In this section we present the translation from GSF to GSF ε (Figure B.7), which inserts ascriptions to ensure that top-level constructors match in every elimination form. We use the following normalization metafunction:

$$\text{norm}(t, G_1, G_2) = \varepsilon t :: G_2, \text{ where } \varepsilon = \mathcal{I}(\Xi, ()G_1, G_2)$$

$$\mathcal{J}(\Xi, ()G_1, G_2) = \mathcal{J}((), \text{lift}_{\Xi}(G_1), \text{lift}_{\Xi}(G_2))$$

Theorem A.31 (Elaboration Preserves Typing) *If $\Delta; \Gamma \vdash t : G$, then $\Delta; \Gamma \vdash t : G \rightsquigarrow t'$ and $t_{\varepsilon} t' G$.*

PROOF. The proof follows by induction on the typing derivation of $\Delta; \Gamma \vdash t : G$. □

$\Delta; \Gamma \vdash v \rightsquigarrow_v u : G$ **Value translation**

$$\begin{array}{c}
\text{(Gb)} \frac{ty(b) = B \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash b \rightsquigarrow_v b : B} \qquad \text{(Gpairu)} \frac{\Delta; \Gamma \vdash v_1 \rightsquigarrow u_1 : G_1 \quad \Delta; \Gamma \vdash v_2 \rightsquigarrow u_2 : G_2}{\Delta; \Gamma \vdash \langle v_1, v_2 \rangle \rightsquigarrow_v \langle u_1, u_2 \rangle : G_1 \times G_2} \\
\text{(G}\lambda\text{)} \frac{\Delta; \Gamma, x : G \vdash t \rightsquigarrow t' : G'}{\Delta; \Gamma \vdash (\lambda x : G.t) \rightsquigarrow_v (\lambda x : G.t') : G \rightarrow G'} \qquad \text{(G}\Lambda\text{)} \frac{\Delta, X; \Gamma \vdash t \rightsquigarrow t' : G \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash (\Lambda X.t) \rightsquigarrow_v (\Lambda X.t') : \forall X.G}
\end{array}$$

$\Delta; \Gamma \vdash t \rightsquigarrow t : G$ **Term translation**

$$\begin{array}{c}
\text{(Gu)} \frac{\Delta; \Gamma \vdash v \rightsquigarrow_v u : G \quad \varepsilon = \mathcal{I}(\cdot, \cdot)G, G)}{\Delta; \Gamma \vdash v \rightsquigarrow \varepsilon u :: G : G} \qquad \text{(Gascu)} \frac{\Delta; \Gamma \vdash v \rightsquigarrow_v u : G \quad \varepsilon = \mathcal{I}(\cdot, \cdot)G, G')}{\Delta; \Gamma \vdash v :: G' \rightsquigarrow \varepsilon u :: G' : G'} \\
\text{(Gx)} \frac{x : G \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x \rightsquigarrow x : G} \qquad \text{(Gasct)} \frac{t \neq v \quad \Delta; \Gamma \vdash t \rightsquigarrow t' : G \quad \varepsilon = \mathcal{I}(\cdot, \cdot)G, G')}{\Delta; \Gamma \vdash t :: G' \rightsquigarrow \varepsilon t' :: G' : G'} \\
\text{(Gpairt)} \frac{(t_1 \neq v_1 \vee t_2 \neq v_2) \quad \Delta; \Gamma \vdash t_1 \rightsquigarrow t'_1 : G_1 \quad \Delta; \Gamma \vdash t_2 \rightsquigarrow t'_2 : G_2}{\Delta; \Gamma \vdash \langle t_1, t_2 \rangle \rightsquigarrow \langle t'_1, t'_2 \rangle : G_1 \times G_2} \\
\text{(Gop)} \frac{\Delta; \Gamma \vdash \bar{t} \rightsquigarrow \bar{t}' : \bar{G}_1 \quad ty(op) = \bar{G}_2 \rightarrow G \quad \bar{t}'' = \overline{norm(t', G_1, G_2)}}{\Delta; \Gamma \vdash op(\bar{t}) \rightsquigarrow op(\bar{t}'') : G} \\
\text{(Gapp)} \frac{\Delta; \Gamma \vdash t_1 \rightsquigarrow t'_1 : G_1 \quad t''_1 = norm(t'_1, G_1, dom^\sharp(G_1) \rightarrow cod^\sharp(G_1)) \quad \Delta; \Gamma \vdash t_2 \rightsquigarrow t'_2 : G_2 \quad t''_2 = norm(t'_2, G_2, dom^\sharp(G_1))}{\Delta; \Gamma \vdash t_1 t_2 \rightsquigarrow t'_1 t'_2 : cod^\sharp(G_2)} \\
\text{(GappG)} \frac{\Delta; \Gamma \vdash t \rightsquigarrow t' : G \quad \Delta \vdash G' \quad t'' = norm(t', G, \forall var^\sharp(G).schm_u^\sharp(G))}{\Delta; \Gamma \vdash t [G'] \rightsquigarrow t'' [G'] : inst^\sharp(G, G')} \\
\text{(Gpairi)} \frac{\Delta; \Gamma \vdash t \rightsquigarrow t' : G \quad t'' = norm(t', G, proj_1^\sharp(G) \times proj_2^\sharp(G))}{\Delta; \Gamma \vdash \pi_i(t) \rightsquigarrow \pi_i(t'') : proj_i^\sharp(G)} \\
\begin{array}{ll}
var^\sharp : \text{GTYPE} \rightarrow \text{GTYPE} & schm_u^\sharp : \text{GTYPE} \rightarrow \text{GTYPE} \\
var^\sharp(\forall X.G) = X & schm_u^\sharp(\forall X.G) = G \\
var^\sharp(?) = X \text{ fresh} & schm_u^\sharp(?) = ? \\
var^\sharp(G) \text{ undefined o/w} & schm_u^\sharp(G) \text{ undefined o/w} \\
norm(t, G_1, G_2) = \varepsilon t :: G_2, \text{ where } \varepsilon = \mathcal{I}(\cdot, \cdot)G_1, G_2
\end{array}
\end{array}$$

Figure A.4: GSF to GSF ε translation.

A.4 GSF: Properties

In this section we present some properties of GSF. Section A.4.1, presents Type Safety and its proof. Section A.4.2, shows the property and proof about static terms do not fail.

A.4.1 Type Safety

In this section we present the proof of type safety for $\text{GSF}\varepsilon$.

We define what it means for a store to be well typed with respect to a term. Informally, all free locations of a term and of the contents of the store must be defined in the domain of that store. Also, the store must preserve types between intrinsic locations and underlying values.

Lemma A.32 (Canonical forms) *Consider a value $\Xi; \cdot; \cdot \vdash v : G$. Then $v = \varepsilon u :: G$, with $\Xi; \cdot; \cdot \vdash u : G'$ and $\varepsilon \Vdash \Xi \vdash G' \sim G$. Furthermore:*

1. *If $G = B$, then $v = \varepsilon_B b :: B$, with $\Xi; \cdot; \cdot \vdash b : B$ and $\varepsilon_B \Vdash \Xi \vdash B \sim B$.*
2. *If $G = G_1 \rightarrow G_2$, then $v = \varepsilon(\lambda x : G'_1. t) :: G_1 \rightarrow G_2$, with $\Xi; \cdot; x : G'_1 \vdash t : G'_2$ and $\varepsilon \Vdash \Xi \vdash G'_1 \rightarrow G'_2 \sim G_1 \rightarrow G_2$.*
3. *If $G = \forall X. G_1$, then $v = \varepsilon(\Lambda X. t) :: \forall X. G_1$, with $\Xi; \Delta, X; \cdot \vdash t : G'_1$ and $\varepsilon \Vdash \Xi \vdash \forall X. G'_1 \sim \forall X. G_1$.*
4. *If $G = G_1 \times G_2$, then $v = \varepsilon\langle u_1, u_2 \rangle :: G_1 \times G_2$, with $\Xi; \cdot; \cdot \vdash u_1 : G'_1$, $\Xi; \cdot; \cdot \vdash u_2 : G'_2$ and $\varepsilon \Vdash \Xi \vdash G'_1 \times G'_2 \sim G_1 \times G_2$.*

PROOF. By direct inspection of the formation rules of evidence augmented terms. □

Lemma A.33 (Substitution) *If $\Xi; \Delta; \Gamma, x : G_1 \vdash t : G$, and $\Xi; \cdot; \cdot \vdash v : G_1$, then $\Xi; \Delta; \Gamma \vdash t[v/x] : G$.*

PROOF. By induction on the derivation of $\Xi; \Delta; \Gamma, x : G_1 \vdash t : G$. □

Lemma A.34 *If $\varepsilon \Vdash \Xi; \Delta, X \vdash G_1 \sim G_2$, $\Xi; \cdot \vdash G'$, $\alpha \notin \text{dom}(\Xi)$, and $E = \text{lift}_{\Xi}(G')$, then $\varepsilon[\alpha^{E'} / X] \Vdash \Xi, \alpha := G'; \Delta \vdash G_1[\alpha / X] \sim G_2[\alpha / X]$.*

PROOF. By induction on the judgment $\varepsilon \Vdash \Xi; \Delta, X \vdash G_1 \sim G_2$ and the definition of evidences. □

Lemma A.35 (Type Substitution) *If $\Xi; \Delta, X; \Gamma \vdash t : G$, $\Xi; \cdot \vdash G'$, $\alpha \notin \text{dom}(\Xi)$, and $E = \text{lift}_{\Xi}(G')$, then $\Xi, \alpha := G'; \Delta; \Gamma \vdash t\alpha^E / X : G[\alpha / X]$.*

PROOF. By induction on the derivation of $\Xi; \Delta, X; \Gamma \vdash t : G$ and Lemma A.34. □

Lemma A.36 *If $\varepsilon_1 \Vdash \Xi; \Delta \vdash G'_1 \sim G_1$, and $\varepsilon_2 \Vdash \Xi; \Delta \vdash G'_2 \sim G_2$, then $\varepsilon_1 \times \varepsilon_2 \Vdash \Xi; \Delta \vdash G'_1 \times G'_2 \sim G_1 \times G_2$.*

PROOF. By definition of the judgment $\varepsilon \Vdash \Xi; \Delta, X \vdash G'_1 \times G'_2 \sim G_1 \times G_2$ and the definition of evidences. \square

Lemma A.37 *If $\varepsilon \Vdash \Xi; \Delta \vdash G' \sim G$ then $p_i(\varepsilon) \Vdash \Xi; \Delta \vdash \text{proj}_i^\sharp(G') \sim \text{proj}_i^\sharp(G)$.*

PROOF. By definition of judgment $\varepsilon \Vdash \Xi; \Delta, X \vdash \text{proj}_i^\sharp(G') \sim \text{proj}_i^\sharp(G)$ and the definition of evidences. \square

Proposition A.38 (\longrightarrow is well defined) *If $\Xi; \cdot; \cdot \vdash t : G$, then either*

- $\Xi \triangleright t \longrightarrow \Xi' \triangleright t', \Xi \subseteq \Xi'$ and $\Xi'; \cdot; \cdot \vdash t' : G$; or
- $\Xi \triangleright t \longrightarrow \mathbf{error}$

PROOF. By induction on the structure of a derivation of $\Xi \triangleright t \longrightarrow r$, considering the last rule used in the derivation.

Case (Rapp). Then $t = (\varepsilon_1(\lambda x : G_{11}.t_1) :: G_1 \rightarrow G_2) (\varepsilon_2 u :: G_1)$. Then

$$(Eapp) \frac{\frac{\frac{\Xi; \cdot; x : G_{11} \vdash t_1 : G_{12}}{\Xi; \cdot; \cdot \vdash (\lambda x : G_{11}.t_1) : G_{11} \rightarrow G_{12}}{\varepsilon_1 \Vdash \Xi; \cdot \vdash G_{11} \rightarrow G_{12} \sim G_1 \rightarrow G_2} (Easc) \quad \frac{\frac{\Xi; \cdot; \cdot \vdash u : G'_2}{\varepsilon_2 \Vdash \Xi; \cdot \vdash G'_2 \sim G_1} (Easc)}{\Xi; \cdot; \cdot \vdash (\varepsilon_2 u :: G_1) : G_1} (Easc)}{\Xi; \cdot; \cdot \vdash (\varepsilon_1(\lambda x : G_{11}.t_1) :: G_1 \rightarrow G_2) : G} (Easc)}{\Xi; \cdot; \cdot \vdash (\varepsilon_1(\lambda x : G_{11}.t_1) :: G) (\varepsilon_2 u :: G_1) : G_2} (Eapp)$$

If $\varepsilon' = (\varepsilon_2; \text{dom}(\varepsilon_1))$ is not defined, then $\Xi \triangleright t \longrightarrow \mathbf{error}$, and then the result hold immediately. Suppose that consistent transitivity does hold, then

$$\Xi \triangleright (\varepsilon_1(\lambda x : G_{11}.t_1) :: G_1 \rightarrow G_2) (\varepsilon_2 u :: G_1) \longrightarrow \Xi \triangleright \text{cod}(\varepsilon_1)(t_1[\varepsilon' u :: G_{11}]/x) :: G_2$$

As $\varepsilon_2 \vdash G'_2 \sim G_1$ and by inversion lemma $\text{dom}(\varepsilon_1) \vdash G_1 \sim G_{11}$, then $\varepsilon' \vdash G'_2 \sim G_{11}$. Therefore $\Xi; \cdot; \cdot \vdash \varepsilon' u :: G_{11} : G_{11}$, and by Lemma A.33, $\Xi; \cdot; \cdot \vdash t[(\varepsilon' u :: G_{11})/x] : G_{12}$.

Let us call $t'' = t[(\varepsilon' u :: G_{11})/x]$. Then

$$(Easc) \frac{\Xi; \cdot; \cdot \vdash t_1[\varepsilon' u :: G_{11}]/x : G_{12} \quad \text{cod}(\varepsilon_1) \Vdash \Xi; \cdot \vdash G_{12} \sim G_2}{\Xi; \cdot; \cdot \vdash \text{cod}(\varepsilon_1)(t_1[\varepsilon' u :: G_{11})/x] :: G_2 : G_2}$$

and the result holds.

Case (RappG). Then $t = (\varepsilon \Lambda X.t_1 :: \forall X.G_x) [G']$. Consider $G_x = \text{schm}_u^\sharp(G)$, then

$$(E_{\text{app}G}) \frac{(E_{\text{asc}}) \frac{\Xi; X; \cdot \vdash t_1 : G_1 \quad \varepsilon \Vdash \Xi; X; \cdot \vdash G_1 \sim \forall X.G_x}{\Xi; ; \cdot \vdash (\varepsilon \Lambda X.t_1 :: \forall X.G_x) : \forall X.G_x}}{\Xi; ; \cdot \vdash (\varepsilon \Lambda X.t_1 :: \forall X.G_x) [G'] : G_x[G'/X]} \quad \Xi; \cdot \vdash G'$$

Then

$$\Xi \triangleright (\varepsilon \Lambda X.t_1 :: G) [G'] \longrightarrow \Xi' \triangleright \varepsilon_G^{E'/\alpha^{E'}} (\varepsilon[\alpha^{E'}] t_1 \alpha^{E'} / X :: G_x[\alpha/X]) :: G_x[G'/X]$$

where $\Xi' \triangleq \Xi, \alpha := G', \alpha \notin \text{dom}(\Xi)$, and $E' \triangleq \text{lift}_\Xi(G')$, and $\varepsilon_{\forall X.G_x}^{E'/\alpha^{E'}} = \langle \text{lift}_\Xi(G_x)[\alpha^{E'}/X], \text{lift}_\Xi(G_x[G'/X]) \rangle$. Notice that $\langle \text{lift}_\Xi(G_x[\alpha/X]), \text{lift}_\Xi(G_x[G'/X]) \rangle = \mathcal{I}(\cdot, \cdot)G_x[\alpha/X], G_x[G'/X]$, and by definition of the special substitution, $\text{lift}_\Xi(G_x)[\alpha^{E'}/X] \sqsubseteq \text{lift}_\Xi(G_x[\alpha/X])$ (because $\text{lift}_\Xi(\alpha) = \alpha^{E'}$, and the substitution on evidences just extend unknowns with α). Therefore $\varepsilon_{\forall X.G_x}^{E'/\alpha^{E'}} \sqsubseteq \mathcal{I}(\cdot, \cdot)G_x[\alpha/X], G_x[G'/X]$, and $\varepsilon_{\forall X.G_x}^{E'/\alpha^{E'}} \Vdash \Xi; \cdot \vdash G_x[\alpha/X] \sim G_x[G'/X]$. Also by Lemma A.34 $\varepsilon[\alpha^{E'}] \Vdash \Xi; \cdot \vdash G_1[\alpha/X] \sim G_x[\alpha/X]$, and by Lemma A.35, $\Xi; ; \cdot \vdash t_1 \alpha^{E'} / X : G_1[\alpha/X]$.

Then, as $\Xi \subseteq \Xi'$,

$$(E_{\text{asc}}) \frac{(E_{\text{asc}}) \frac{\Xi; ; \cdot \vdash t_1 \alpha^{E'} / X : G_1[\alpha/X] \quad \varepsilon[\alpha^{E'}] \Vdash \Xi; \cdot \vdash G_1[\alpha/X] \sim G_x[\alpha/X]}{\Xi; ; \cdot \vdash (\varepsilon[\alpha^{E'}] t_1 \alpha^{E'} / X :: G_x[\alpha/X]) : G_x[\alpha/X]} \quad \varepsilon_G^{E'/\alpha^{E'}} \Vdash \Xi; \cdot \vdash G_x[\alpha/X] \sim G_x[G'/X]}{\Xi; ; \cdot \vdash \varepsilon_G^{E'/\alpha^{E'}} (\varepsilon[\alpha^{E'}] t_1 \alpha^{E'} / X :: G_x[\alpha/X]) :: G_x[G'/X] : G_x[G'/X]}$$

and the result holds.

Case (Rasc). Then $t = \varepsilon_1(\varepsilon_2 u :: G_2) :: G$. Then

$$(E_{\text{asc}}) \frac{(E_{\text{asc}}) \frac{\Xi; ; \cdot \vdash u : G_u \quad \varepsilon_2 \Vdash \Xi; \cdot \vdash G_u \sim G_2}{\Xi; ; \cdot \vdash \varepsilon_2 u :: G_2 : G_2}}{\Xi; ; \cdot \vdash \varepsilon_1(\varepsilon_2 u :: G_2) :: G : G} \quad \varepsilon_1 \Vdash \Xi; \cdot \vdash G_2 \sim G$$

If $(\varepsilon_2 \circledast \varepsilon_1)$ is not defined, then $\Xi \triangleright t \longrightarrow \mathbf{error}$, and then the result hold immediately. Suppose that consistent transitivity does hold, then

$$\Xi \triangleright \varepsilon_1(\varepsilon_2 u :: G_2) :: G \longrightarrow \Xi \triangleright (\varepsilon_2 \circledast \varepsilon_1)u :: G$$

where $(\varepsilon_2 \circledast \varepsilon_1) \Vdash \Xi; \cdot \vdash G_u \sim G$. Then

$$(E_{\text{asc}}) \frac{\Xi; ; \cdot \vdash u : G_u \quad (\varepsilon_2 \circledast \varepsilon_1) \Vdash \Xi; \cdot \vdash G_u \sim G}{\Xi; ; \cdot \vdash (\varepsilon_2 \circledast \varepsilon_1)u :: G : G}$$

and the result follows.

Case (Rop). Then $t = \text{op}(\overline{\varepsilon u :: B'})$. Then

$$(E_{\text{op}}) \frac{(E_{\text{asc}}) \frac{\overline{\Xi; ; \cdot \vdash u : G_u} \quad \overline{\varepsilon \Vdash \Xi; \cdot \vdash G_u \sim B'}}{\Xi; \Delta; \Gamma \vdash \overline{\varepsilon u :: B'} : \overline{B'}} \quad \text{ty}(\text{op}) = \overline{B'} \rightarrow B}{\Xi; ; \cdot \vdash \text{op}(\overline{\varepsilon u :: B'}) : B}$$

Let us assume that $ty(op) : \overline{B'} \rightarrow B$.

$$\Xi \triangleright op(\overline{\varepsilon u :: B'}) \longrightarrow \Xi \triangleright \varepsilon_B \delta(op, \bar{u}) :: B$$

But as $\varepsilon_B \vdash \Xi; \cdot \vdash B \sim B$, then

$$(Easc) \frac{\Xi; \cdot; \cdot \vdash \delta(op, \bar{u}) : B \quad \varepsilon_B \Vdash \Xi; \cdot \vdash B \sim B}{\Xi; \cdot; \cdot \vdash \varepsilon_B \delta(op, \bar{u}) :: B : B}$$

and the result follows.

Case (Rpair). Then $t = \langle \varepsilon_1 u_1 :: G_1, \varepsilon_2 u_2 :: G_2 \rangle$. Then

$$(Epair) \frac{(Easc) \frac{\Xi; \cdot; \cdot \vdash u_1 : G'_1}{\varepsilon_1 \Vdash \Xi; \cdot \vdash G'_1 \sim G_1} \quad (Easc) \frac{\Xi; \cdot; \cdot \vdash u_2 : G'_2}{\varepsilon_2 \Vdash \Xi; \cdot \vdash G'_2 \sim G_2}}{\Xi; \cdot; \cdot \vdash \langle \varepsilon_1 u_1 :: G_1, \varepsilon_2 u_2 :: G_2 \rangle : G_1 \times G_2}$$

Then

$$\Xi \triangleright \langle \varepsilon_1 u_1 :: G_1, \varepsilon_2 u_2 :: G_2 \rangle \longrightarrow \Xi \triangleright (\varepsilon_1 \times \varepsilon_2) \langle u_1, u_2 \rangle :: G_1 \times G_2$$

By Lemma B.6, $\varepsilon_1 \times \varepsilon_2 \Vdash \Xi; \cdot \vdash G'_1 \times G'_2 \sim G_1 \times G_2$. Then

$$(Easc) \frac{(Epair) \frac{\Xi; \cdot; \cdot \vdash u_1 : G'_1 \quad \Xi; \cdot; \cdot \vdash u_2 : G'_2}{\Xi; \cdot; \cdot \vdash \langle u_1, u_2 \rangle : G'_1 \times G'_2} \quad \varepsilon_1 \times \varepsilon_2 \Vdash \Xi; \cdot \vdash G'_1 \times G'_2 \sim G_1 \times G_2}{\Xi; \cdot; \cdot \vdash (\varepsilon_1 \times \varepsilon_2) \langle u_1, u_2 \rangle :: G_1 \times G_2 : G_1 \times G_2}$$

and the result holds.

Case (Rproj_i). Then $t = \pi_i(\varepsilon \langle u_1, u_2 \rangle :: G)$. Then

$$(Epairi) \frac{(Easc) \frac{\Xi; \cdot; \cdot \vdash u_i : G'_i}{\Xi; \cdot; \cdot \vdash \langle u_1, u_2 \rangle : G'_1 \times G'_2} \quad \varepsilon \Vdash \Xi; \cdot \vdash G'_1 \times G'_2 \sim G}{\Xi; \cdot; \cdot \vdash \varepsilon \langle u_1, u_2 \rangle :: G}}{\Xi; \cdot; \cdot \vdash \pi_i(\varepsilon \langle u_1, u_2 \rangle :: G) : proj_i^\sharp(G)}$$

Then

$$\Xi \triangleright \pi_i(\varepsilon \langle u_1, u_2 \rangle :: G) \longrightarrow \Xi \triangleright p_i(\varepsilon) u_i :: proj_i^\sharp(G)$$

By Lemma B.7, $p_i(\varepsilon) \Vdash \Xi; \cdot \vdash proj_i^\sharp(G'_1 \times G'_2) \sim proj_i^\sharp(G)$. Then

$$(Easc) \frac{\Xi; \cdot; \cdot \vdash u_i : G'_i \quad p_i(\varepsilon) \Vdash \Xi; \cdot \vdash proj_i^\sharp(G'_1 \times G'_2) \sim proj_i^\sharp(G)}{\Xi; \cdot; \cdot \vdash p_i(\varepsilon) u_i :: proj_i^\sharp(G) : proj_i^\sharp(G)}$$

and the result holds. □

Proposition A.39 (\mapsto is well defined) *If $\Xi; \cdot \vdash t : G$, then either*

- $\Xi \triangleright t \mapsto \Xi' \triangleright t'$, $\Xi \subseteq \Xi'$ and $\Xi'; \cdot \vdash t' : G$; or
- $\Xi \triangleright t \mapsto \mathbf{error}$

PROOF. By induction on the structure of t .

- If t has some of this form: $\varepsilon_2(\varepsilon_1 u :: G_1) :: G_2$, $op(\overline{\varepsilon u :: G})$, $(\lambda x : G_{11}.t) :: G_1 \rightarrow G_2$ ($\varepsilon_2 u :: G_1$), $\langle \varepsilon_1 u_1 :: G_1, \varepsilon_2 u_2 :: G_2 \rangle$, $\pi_i(\varepsilon \langle u_1, u_2 \rangle :: G_1 \times G_2)$ or $(\varepsilon \Lambda X.t :: \forall X.G) [G']$, then by well-definedness of \mapsto (Prop A.38), $\Xi \triangleright t \mapsto \Xi' \triangleright t'$ and $\Xi \subseteq \Xi'$ and $\Xi'; \cdot \vdash t' : G$ or $\Xi \triangleright t \mapsto \mathbf{error}$, .
 If $\Xi \triangleright t \mapsto \Xi' \triangleright t'$, $\Xi \subseteq \Xi'$ and $\Xi'; \cdot \vdash t' : G$, then by the rule $R\mapsto$ the result holds.
 If $\Xi \triangleright t \mapsto \mathbf{error}$, then by the rule $Rerr$ $\Xi \triangleright t \mapsto \mathbf{error}$ and the result holds immediately.
- If $t = f[t_1]$, we know that $\Xi; \cdot \vdash f[t_1] : G$ and $\Xi; \cdot \vdash t_1 : G'$, where $f : G' \rightarrow G$. Then, by the induction hypothesis $\Xi \triangleright t_1 \mapsto \Xi' \triangleright t'_1$, $\Xi \subseteq \Xi'$ and $\Xi'; \cdot \vdash t'_1 : G$ or $\Xi \triangleright t_1 \mapsto \Xi' \triangleright \mathbf{error}$.
 If $\Xi \triangleright t_1 \mapsto \Xi' \triangleright t'_1$, by the Rf rule the result holds.
 If $\Xi \triangleright t_1 \mapsto \Xi' \triangleright \mathbf{error}$, by the $Rferr$ rule the result holds. .

□

Proposition A.40 (\mapsto is well defined) *If $\Xi; \cdot \vdash t : G$, $t \rightsquigarrow t_\varepsilon$, then t_ε is a value v ; or $\Xi \triangleright t_\varepsilon \mapsto \Xi' \triangleright t'_\varepsilon$, $\Xi \subseteq \Xi'$ and $\Xi'; \cdot \vdash t'_\varepsilon : G$; or $\Xi \triangleright t_\varepsilon \mapsto \mathbf{error}$.*

PROOF. By induction on the structure of t , using Lemma A.39 and Canonical Forms (Lemma B.5). □

Now we can establish type safety of GSF: programs of GSF do not get stuck, though they may terminate with cast errors. Also the store of a program is well typed.

Proposition A.41 (Type Safety) *If $\vdash t : G$ then either $t \Downarrow \Xi \triangleright v$ with $\Xi \triangleright v : G$, $t \Downarrow \mathbf{error}$, or $t \Uparrow$.*

PROOF. Direct by A.40. □

A.4.2 Static Terms Do Not Fail

Lemma A.42 (*Properties of consistent transitivity*).

- (a) *Associativity.* $(\varepsilon_1 \circledast \varepsilon_2) \circledast \varepsilon_3 = \varepsilon_1 \circledast (\varepsilon_2 \circledast \varepsilon_3)$, or both are undefined.
- (b) *Optimality.* If $\varepsilon = \varepsilon_1 \circledast \varepsilon_2$ is defined, then $\pi_1(\varepsilon) \sqsubseteq \pi_1(\varepsilon_1)$ and $\pi_2(\varepsilon) \sqsubseteq \pi_2(\varepsilon_2)$.
- (c) *Monotonicity.* If $\varepsilon_1 \sqsubseteq \varepsilon'_1$ and $\varepsilon_2 \sqsubseteq \varepsilon'_2$ and $\varepsilon_1 \circledast \varepsilon_2$ is defined, then $\varepsilon'_1 \circledast \varepsilon'_2$ is defined and $\varepsilon_1 \circledast \varepsilon_2 \sqsubseteq \varepsilon'_1 \circledast \varepsilon'_2$.

PROOF. A direct result of the application of the AGT framework. □

Lemma A.43 *If ε_1 and ε_2 two static evidences, such that $\varepsilon_1 \Vdash \Xi; \Delta \vdash T_1 \sim T_2$ and $\varepsilon_2 \Vdash \Xi; \Delta \vdash T_2 \sim T_3$, then $\varepsilon_1 \mathbin{\text{;}} \varepsilon_2 = \langle p_1(\varepsilon_1), p_2(\varepsilon_2) \rangle$.*

PROOF. Straightforward induction on types T_1, T_2, T_3 ($\Xi; \Delta \vdash T_2 \sim T_3$ coincides with $\Xi; \Delta \vdash T_2 = T_3$), and optimality of evidences (Lemma 3.14), because the resulting evidence cannot gain precision as each component of the evidences are static (note that precision $\cdot \sqsubseteq \cdot$ between static types coincide with equality of static types $\Xi; \Delta \vdash \cdot = \cdot$). \square

Lemma A.44 *Let T_1 and T_2 two static types, and Ξ a static store, such that $\Xi; \Delta \vdash T_1 \sim T_2$. Then $\mathcal{G}(\cdot, \cdot)T_1, T_2 = \mathcal{G}(\cdot, \cdot)\text{lift}_{\Xi}(T_1), \text{lift}_{\Xi}(T_2) = \langle \text{lift}_{\Xi}(T_1), \text{lift}_{\Xi}(T_2) \rangle$.*

PROOF. Straightforward induction on types T_1, T_2 , and noticing that we cannot gain precision from the types. \square

Proposition A.45 (Static terms progress and Preservation) *Let t be a static term, Ξ a static store ($\Xi = \Sigma$), and G a static type ($G = T$). If $\Sigma; \cdot; \cdot \vdash t : T$, then either $\Sigma \triangleright t \mapsto \Sigma' \triangleright t'$ and $\Sigma'; \cdot; \cdot \vdash t' : T$, for some Σ' and t' static; or t is a value v .*

PROOF. By induction on the structure of a derivation of $\Sigma; \cdot; \cdot \vdash t : T$.

Note that $\Xi; \Delta \vdash T_1 \sim T_2$ coincides with $\Xi; \Delta \vdash T_1 = T_2$, so we use the latter notation throughout the proof.

Case ($t = \varepsilon u :: G$). The result is trivial as t is a value.

Case ($t = (\varepsilon_1(\lambda x : T_{11}.t_1) :: T_1 \rightarrow T_2) (\varepsilon_2 u :: T_1)$). Then

$$(Eapp) \frac{\begin{array}{c} \Xi; \cdot; x : T_{11} \vdash t_1 : T_{12} \\ \hline \Xi; \cdot; \cdot \vdash (\lambda x : T_{11}.t_1) : T_{11} \rightarrow T_{12} \\ \varepsilon_1 \Vdash \Sigma; \Delta \vdash T_{11} \rightarrow T_{12} = T_1 \rightarrow T_2 \end{array} \quad \begin{array}{c} \Xi; \cdot; \cdot \vdash u : T'_2 \\ \hline \varepsilon_2 \Vdash \Sigma; \Delta \vdash T'_2 = T_1 \\ \Xi; \cdot; \cdot \vdash (\varepsilon_2 u :: T_1) : T_1 \end{array}}{\Xi; \cdot; \cdot \vdash (\varepsilon_1(\lambda x : T_{11}.t_1) :: T_1 \rightarrow T_2) (\varepsilon_2 u :: T_1) : T_2}$$

By Lemma A.43, $\varepsilon' = (\varepsilon_2 \mathbin{\text{;}} \text{dom}(\varepsilon_1))$ is defined and by Lemma A.44, the new evidence is also static. Then

$$\Xi \triangleright (\varepsilon_1(\lambda x : T_{11}.t_1) :: T) (\varepsilon_2 u :: T_1) \longrightarrow \Xi \triangleright \text{cod}(\varepsilon_1)(t_1[\varepsilon' u :: T_{11}]/x) :: T_2$$

And the result holds immediately by the Lemma A.33 and the typing rule (Easc).

Case ($t = (\varepsilon \Lambda X.t_1 :: \forall X.T_x) [T']$). Then

$$(EappT) \frac{\begin{array}{c} \Xi; X; \cdot \vdash t_1 : T_1 \quad \varepsilon \Vdash \Sigma; \Delta \vdash [= \Xi; X; \cdot]T_1 \forall X.T_x \\ \hline \Xi; \cdot; \cdot \vdash (\varepsilon \Lambda X.t_1 :: \forall X.T_x) : T \end{array} \quad \Xi; \cdot \vdash T'}{\Xi; \cdot; \cdot \vdash (\varepsilon \Lambda X.t_1 :: \forall X.T_x) [T'] : T_x[T'/X]}$$

Then

$$(\varepsilon \Lambda X.t_1 :: \forall X.T_x) [T'] \longrightarrow \Xi' \triangleright \varepsilon_{\forall X.T_x}^{E'/\alpha^{E'}} (\varepsilon[\alpha^{E'}]t_1 \alpha^{E'} / X :: T_x[\alpha / X]) :: T_x[T' / X]$$

where $\Xi' \triangleq \Xi, \alpha := T', \alpha \notin \text{dom}(\Xi)$, and $E' \triangleq \text{lift}_{\Xi}(T')$, and

$\varepsilon_{\forall X.T_x}^{E'/\alpha^{E'}} = \langle \text{lift}_{\Xi}(T_x)[\alpha^{E'} / X], \text{lift}_{\Xi}(T_x[T' / X]) \rangle$. Then, $\Xi \subseteq \Xi'$, and Ξ' is extended with a type name that maps to a static type. Finally, the result holds immediately by the Lemma A.35 and Lemma A.34, and the typing rule (Easc).

Case ($t = \Xi \triangleright \varepsilon_1(\varepsilon_2 u :: T_2) :: T$). Then

$$(Easc) \frac{(Easc) \frac{\Xi; \cdot; \cdot \vdash u : T_u \quad \varepsilon_2 \Vdash \Sigma; \Delta \vdash T_u = T_2}{\Xi; \cdot; \cdot \vdash \varepsilon_2 u :: T_2 : T_2} \quad \varepsilon_1 \Vdash \Sigma; \Delta \vdash T_2 = T}{\Xi; \cdot; \cdot \vdash \varepsilon_1(\varepsilon_2 u :: T_2) :: T : T}}$$

By Lemma A.43, $\varepsilon_2 \circ \varepsilon_1$ is defined and by Lemma A.44, the new evidence is also static. Then

$$\Xi \triangleright \varepsilon_1(\varepsilon_2 u :: T_2) :: T \longrightarrow \Xi \triangleright (\varepsilon_2 \circ \varepsilon_1)u :: T$$

and the result holds by the typing rule (Easc).

Case ($t = \text{op}(\overline{\varepsilon u :: B'})$). Then

$$(Easc) \frac{(Easc) \frac{\overline{\Xi; \cdot; \cdot \vdash u : T_u} \quad \overline{\varepsilon \Vdash \Sigma; \Delta \vdash T_u = B'}}{\Xi; \Delta; \Gamma \vdash \overline{\varepsilon u} :: B' : \overline{B'}} \quad \text{ty}(\text{op}) = \overline{B'} \rightarrow B}{\Xi; \cdot; \cdot \vdash \text{op}(\overline{\varepsilon u :: B'}) : B}}$$

Let us assume that $\text{ty}(\text{op}) : \overline{B'} \rightarrow B$. Then

$$\Xi \triangleright \text{op}(\overline{\varepsilon u :: B'}) \longrightarrow \Xi \triangleright \varepsilon_B \delta(\text{op}, \overline{u}) :: B$$

And the result holds by the typing rule (Easc).

Case ($t = \langle \varepsilon_1 u_1 :: T_1, \varepsilon_2 u_2 :: T_2 \rangle$). Then

$$(Epair) \frac{(Easc) \frac{\Xi; \cdot; \cdot \vdash u_1 : T'_1 \quad \varepsilon_1 \Vdash \Sigma; \Delta \vdash T'_1 = T_1}{\Xi; \cdot; \cdot \vdash \varepsilon_1 u_1 :: T_1} \quad (Easc) \frac{\Xi; \cdot; \cdot \vdash u_2 : T'_2 \quad \varepsilon_2 \Vdash \Sigma; \Delta \vdash T'_2 = T_2}{\Xi; \cdot; \cdot \vdash \varepsilon_2 u_2 :: T_2}}{\Xi; \cdot; \cdot \vdash \langle \varepsilon_1 u_1 :: T_1, \varepsilon_2 u_2 :: T_2 \rangle : T_1 \times T_2}}$$

Then

$$\Xi \triangleright \langle \varepsilon_1 u_1 :: T_1, \varepsilon_2 u_2 :: T_2 \rangle \longrightarrow \Xi \triangleright (\varepsilon_1 \times \varepsilon_2) \langle u_1, u_2 \rangle :: T_1 \times T_2$$

and the result holds by the Lemma B.6.

Case ($t = \pi_i(\varepsilon \langle u_1, u_2 \rangle :: T)$). Then

$$(Epair) \frac{(Easc) \frac{\Xi; \cdot; \cdot \vdash u_i : T'_i}{\Xi; \cdot; \cdot \vdash \langle u_1, u_2 \rangle : T'_1 \times T'_2} \quad \varepsilon \Vdash \Sigma; \Delta \vdash T'_1 \times T'_2 = T}{\varepsilon \langle u_1, u_2 \rangle :: T}}{\Xi; \cdot; \cdot \vdash \pi_i(\varepsilon \langle u_1, u_2 \rangle :: T) : \text{proj}_i^\sharp(T)}$$

Then

$$\Xi \triangleright \pi_i(\varepsilon \langle u_1, u_2 \rangle :: T) \longrightarrow \Xi \triangleright p_i(\varepsilon) u_i :: \text{proj}_i^\sharp(T)$$

And the result holds by Lemma B.7.

Case $(t = t_1 t_2)$. Then by induction hypothesis $\Xi \triangleright t_1 \mapsto \Xi \triangleright t'_1$, and t'_1 is static, and so $t'_1 t_2$.

Case $(t = v t_2)$. Then by induction hypothesis $\Xi \triangleright t_2 \mapsto \Xi \triangleright t'_2$, and t'_2 is static, and so $v t'_2$.

Case $(t = t_1[T], t = \langle t_1, t_2 \rangle, t = \text{op}(\overline{t_1}), t = \pi_i(t_1))$. Similar inductive reasoning to application cases.

□

Proposition A.46 (Static terms do not fail) *Let t be a static term. If $\vdash t : T$ then $\neg(t \Downarrow \mathbf{error})$.*

PROOF. Direct by Lemma A.45.

□

A.5 GSF and The Dynamic Gradual Guarantee

In this section, we prove the weaker variant of the DGG in GSF ε and then in GSF. We also present auxiliary definitions and Propositions.

A.5.1 Evidence Type Precision

This section show the definition of evidence type precision.

$$\begin{array}{c}
 \frac{}{B \leq B} \quad \frac{}{X \leq X} \quad \frac{}{\alpha \leq \alpha} \quad \frac{}{B \leq ?} \quad \frac{E_1 \rightarrow E_2 \leq ? \rightarrow ?}{E_1 \rightarrow E_2 \leq ?} \quad \frac{}{? \leq ?} \\
 \\
 \frac{E_1 \leq E_3 \quad E_2 \leq E_4}{E_1 \rightarrow E_2 \leq E_3 \rightarrow E_4} \quad \frac{E_1 \leq E_2}{\forall X.E_1 \leq \forall X.E_2} \quad \frac{E_1 \leq E_2}{\alpha^{E_1} \leq \alpha^{E_2}} \quad \frac{E_1 \leq E_3 \quad E_2 \leq E_4}{\langle E_1, E_2 \rangle \leq \langle E_3, E_4 \rangle}
 \end{array}$$

A.5.2 Monotonicity of Evidence Transitivity and Instantiation

This section presents the proofs of the monotonicity of evidence transitivity and instantiation proposition.

Proposition A.47 (\leq -Monotonicity of Consistent Transitivity) *If $\varepsilon_1 \leq \varepsilon_2$, $\varepsilon_3 \leq \varepsilon_4$, and $\varepsilon_1 \circ \varepsilon_3$ is defined, then $\varepsilon_1 \circ \varepsilon_3 \leq \varepsilon_2 \circ \varepsilon_4$.*

PROOF. By definition of consistent transitivity for $=$ and the definition of precision.

Case ($\varepsilon_i = \langle B, B \rangle$). The results follows immediately, due

$$\langle B, B \rangle = (\langle B, B \rangle \circ \langle B, B \rangle) \leq ((\langle B, B \rangle \circ \langle B, B \rangle) = \langle B, B \rangle)$$

Case ($[X]$ - $\varepsilon_i = \langle X, X \rangle$). The results follows immediately, due

$$\langle X, X \rangle = (\langle X, X \rangle \circ \langle X, X \rangle) \leq ((\langle X, X \rangle \circ \langle X, X \rangle) = \langle X, X \rangle)$$

Case ($[\alpha_1]$ - $\varepsilon_1 = \langle \alpha^{E_1}, E'_1 \rangle, \varepsilon_2 = \langle \alpha^{E_2}, E'_2 \rangle, \varepsilon_3 = \langle E_3, E'_3 \rangle, \varepsilon_4 = \langle E_4, E'_4 \rangle$). By the definition of \leq , we know that $\langle E_1, E'_1 \rangle \leq \langle E_2, E'_2 \rangle$ and $\langle E_3, E'_3 \rangle \leq \langle E_4, E'_4 \rangle$. By the definition of transitivity we know that $\langle \alpha^{E_1}, E'_1 \rangle \circ \langle E_3, E'_3 \rangle = \langle \alpha^{E_5}, E'_5 \rangle$ and $\langle \alpha^{E_2}, E'_2 \rangle \circ \langle E_4, E'_4 \rangle = \langle \alpha^{E_6}, E'_6 \rangle$, where $\langle E_5, E'_5 \rangle = \langle E_1, E'_1 \rangle \circ \langle E_3, E'_3 \rangle$ and $\langle E_6, E'_6 \rangle = \langle E_2, E'_2 \rangle \circ \langle E_4, E'_4 \rangle$. Therefore, we are required to prove that $\langle \alpha^{E_5}, E'_5 \rangle \leq \langle \alpha^{E_6}, E'_6 \rangle$, or what is the same $\langle E_5, E'_5 \rangle \leq \langle E_6, E'_6 \rangle$. But the result follows immediately by the induction hypothesis on $\langle E_1, E'_1 \rangle \leq \langle E_2, E'_2 \rangle$ and $\langle E_3, E'_3 \rangle \leq \langle E_4, E'_4 \rangle$.

Case ($[\alpha_2]$ - $\varepsilon_1 = \langle E_1, \alpha^{E'_1} \rangle, \varepsilon_2 = \langle E_2, \alpha^{E'_2} \rangle, \varepsilon_3 = \langle \alpha^{E_3}, E'_3 \rangle, \varepsilon_4 = \langle \alpha^{E_4}, E'_4 \rangle$). By the definition of \leq , we know that $\langle E_1, \alpha^{E'_1} \rangle \leq \langle E_2, \alpha^{E'_2} \rangle$ and $\langle \alpha^{E_3}, E'_3 \rangle \leq \langle \alpha^{E_4}, E'_4 \rangle$. By the definition of transitivity we know that $\langle E_1, \alpha^{E'_1} \rangle \circ \langle \alpha^{E_3}, E'_3 \rangle = \langle E_5, E'_5 \rangle$ and $\langle E_2, \alpha^{E'_2} \rangle \circ \langle \alpha^{E_4}, E'_4 \rangle = \langle E_6, E'_6 \rangle$, where $\langle E_5, E'_5 \rangle = \langle E_1, \alpha^{E'_1} \rangle \circ \langle \alpha^{E_3}, E'_3 \rangle$ and $\langle E_6, E'_6 \rangle = \langle E_2, \alpha^{E'_2} \rangle \circ \langle \alpha^{E_4}, E'_4 \rangle$. Therefore, we are required to prove that $\langle E_5, E'_5 \rangle \leq \langle E_6, E'_6 \rangle$. But the result follows immediately by the induction hypothesis on $\langle E_1, \alpha^{E'_1} \rangle \leq \langle E_2, \alpha^{E'_2} \rangle$ and $\langle \alpha^{E_3}, E'_3 \rangle \leq \langle \alpha^{E_4}, E'_4 \rangle$.

Case ($[\alpha_3]$ - $\varepsilon_1 = \langle E_1, E'_1 \rangle, \varepsilon_2 = \langle E_2, E'_2 \rangle, \varepsilon_3 = \langle E_3, \alpha^{E'_3} \rangle, \varepsilon_4 = \langle E_4, \alpha^{E'_4} \rangle$). By the definition of \leq , we know that $\langle E_1, E'_1 \rangle \leq \langle E_2, E'_2 \rangle$ and $\langle E_3, E'_3 \rangle \leq \langle E_4, E'_4 \rangle$. By the definition of transitivity we know that $\langle E_1, E'_1 \rangle \circ \langle E_3, \alpha^{E'_3} \rangle = \langle E_5, \alpha^{E'_5} \rangle$ and $\langle E_2, E'_2 \rangle \circ \langle E_4, \alpha^{E'_4} \rangle = \langle E_6, \alpha^{E'_6} \rangle$, where $\langle E_5, E'_5 \rangle = \langle E_1, E'_1 \rangle \circ \langle E_3, E'_3 \rangle$ and $\langle E_6, E'_6 \rangle = \langle E_2, E'_2 \rangle \circ \langle E_4, E'_4 \rangle$. Therefore, we are required to prove that $\langle E_5, \alpha^{E'_5} \rangle \leq \langle E_6, \alpha^{E'_6} \rangle$, or what is the same $\langle E_5, E'_5 \rangle \leq \langle E_6, E'_6 \rangle$. But the result follows immediately by the induction hypothesis on $\langle E_1, E'_1 \rangle \leq \langle E_2, E'_2 \rangle$ and $\langle E_3, E'_3 \rangle \leq \langle E_4, E'_4 \rangle$.

Case ($[\forall]$ - $\varepsilon_i = \langle \forall X.E_i, \forall X.E'_i \rangle$). By the definition of \leq , we know that $\langle E_1, E'_1 \rangle \leq \langle E_2, E'_2 \rangle$ and $\langle E_3, E'_3 \rangle \leq \langle E_4, E'_4 \rangle$. By the definition of transitivity we know that $\langle \forall X.E_1, \forall X.E'_1 \rangle \circ \langle \forall X.E_3, \forall X.E'_3 \rangle = \langle \forall X.E_5, \forall X.E'_5 \rangle$ and $\langle \forall X.E_2, \forall X.E'_2 \rangle \circ \langle \forall X.E_4, \forall X.E'_4 \rangle = \langle \forall X.E_6, \forall X.E'_6 \rangle$, where $\langle E_5, E'_5 \rangle = \langle E_1, E'_1 \rangle \circ \langle E_3, E'_3 \rangle$ and $\langle E_6, E'_6 \rangle = \langle E_2, E'_2 \rangle \circ \langle E_4, E'_4 \rangle$. Therefore, we are required to prove that $\langle E_5, E'_5 \rangle \leq \langle E_6, E'_6 \rangle$. But the result follows immediately by the induction hypothesis on $\langle E_1, E'_1 \rangle \leq \langle E_2, E'_2 \rangle$ and $\langle E_3, E'_3 \rangle \leq \langle E_4, E'_4 \rangle$.

Case ($[\rightarrow]$ - $\varepsilon_i = \langle E_{1i} \rightarrow E_{2i}, E'_{1i} \rightarrow E'_{2i} \rangle$). By the definition of \leq , we know that $\langle E_{11}, E'_{11} \rangle \leq \langle E_{12}, E'_{12} \rangle$, $\langle E_{13}, E'_{13} \rangle \leq \langle E_{14}, E'_{14} \rangle$, $\langle E_{21}, E'_{21} \rangle \leq \langle E_{22}, E'_{22} \rangle$ and $\langle E_{23}, E'_{23} \rangle \leq \langle E_{24}, E'_{24} \rangle$. By the definition of transitivity we know that $\langle E_{11} \rightarrow E_{21}, E'_{11} \rightarrow E'_{21} \rangle \circ \langle E_{13} \rightarrow E_{23}, E'_{13} \rightarrow E'_{23} \rangle = \langle E_{15} \rightarrow E_{25}, E'_{15} \rightarrow E'_{25} \rangle$ and $\langle E_{12} \rightarrow E_{22}, E'_{12} \rightarrow E'_{22} \rangle \circ \langle E_{14} \rightarrow E_{24}, E'_{14} \rightarrow E'_{24} \rangle = \langle E_{16} \rightarrow E_{26}, E'_{16} \rightarrow E'_{26} \rangle$, where $\langle E'_{15}, E_{15} \rangle = \langle E'_{13}, E_{13} \rangle \circ \langle E'_{11}, E_{11} \rangle$, $\langle E'_{25}, E'_{25} \rangle = \langle E'_{21}, E'_{21} \rangle \circ \langle E'_{24}, E'_{24} \rangle$, $\langle E'_{16}, E'_{16} \rangle = \langle E'_{41}, E_{41} \rangle \circ \langle E'_{12}, E_{12} \rangle$ and $\langle E'_{26}, E'_{26} \rangle = \langle E'_{22}, E'_{22} \rangle \circ \langle E'_{24}, E'_{24} \rangle$.

Therefore, we are required to prove that

$$\langle E_{15} \rightarrow E_{25}, E'_{15} \rightarrow E'_{25} \rangle \leq \langle E_{16} \rightarrow E_{26}, E'_{16} \rightarrow E'_{26} \rangle$$

or what is the same

$$\langle E'_{13}, E_{13} \rangle \circ \langle E'_{11}, E_{11} \rangle = \langle E'_{15}, E_{15} \rangle \leq \langle E'_{16}, E_{16} \rangle = \langle E'_{41}, E_{41} \rangle \circ \langle E'_{12}, E_{12} \rangle$$

and

$$\langle E_{21}, E'_{21} \rangle \circ \langle E_{23}, E'_{23} \rangle = \langle E_{25}, E'_{25} \rangle \leq \langle E_{26}, E'_{26} \rangle = \langle E_{22}, E'_{22} \rangle \circ \langle E_{24}, E'_{24} \rangle$$

But the result follows immediately by the induction hypothesis on $\langle E_{11}, E'_{11} \rangle \leq \langle E_{12}, E'_{12} \rangle$ and $\langle E_{13}, E'_{13} \rangle \leq \langle E_{14}, E'_{14} \rangle$, $\langle E_{21}, E'_{21} \rangle \leq \langle E_{22}, E'_{22} \rangle$ and $\langle E_{23}, E'_{23} \rangle \leq \langle E_{24}, E'_{24} \rangle$.

Case ($[\times]$ - $\varepsilon_i = \langle E_{1i} \times E_{2i}, E'_{1i} \times E'_{2i} \rangle$). Similar to Case $[\rightarrow]$.

Case ($[?_1]$ - $\varepsilon_1 = \langle ?, ? \rangle$). Since $\varepsilon_1 \leq \varepsilon_2$, we know that $\varepsilon_2 = \langle ?, ? \rangle$. Therefore, by the transitivity rules, we know that $\varepsilon_1 \circ \varepsilon_3 = \varepsilon_3$ and $\varepsilon_2 \circ \varepsilon_4 = \varepsilon_4$. Thus, we are required to prove that $\varepsilon_3 \leq \varepsilon_4$, but the result follows immediately by premise.

Case ($[?_2]$ - $\varepsilon_2 = \langle ?, ? \rangle$). The proof follows from some of the previous cases.

- ($\varepsilon_1 = \langle ?, ? \rangle$). The results follows immediately, since it was discussed in Case $[?_1]$.
- ($\varepsilon_3 = \langle ?, ? \rangle$). The results follows immediately, since it was discussed in Case $[?_3]$.
- ($\varepsilon_4 = \langle ?, ? \rangle$). The results follows immediately, since $\varepsilon_1 \circ \varepsilon_3 \leq \langle ?, ? \rangle \circ \langle ?, ? \rangle = \langle ?, ? \rangle$.
- ($\varepsilon_i = \langle B, B \rangle$). The results follows immediately, since $\langle B, B \rangle \circ \langle B, B \rangle \leq \langle ?, ? \rangle \circ \langle B, B \rangle$.
- ($\varepsilon_i = \langle X, X \rangle$). This case is not possible, since $\langle X, X \rangle \not\leq \langle ?, ? \rangle$.
- Case $[\alpha_1]$ ($\varepsilon_1 = \langle E_1, \alpha^{E'_1} \rangle, \varepsilon_2 = \langle ?, ? \rangle, \varepsilon_3 = \langle \alpha^{E'_3}, E'_3 \rangle, \varepsilon_4 = \langle \alpha^{E'_4}, E'_4 \rangle$). This case is not possible, since $\langle \alpha^{E'_1}, E'_1 \rangle \not\leq \langle ?, ? \rangle$.
- Case $[\alpha_2]$ ($\varepsilon_1 = \langle E_1, \alpha^{E'_1} \rangle, \varepsilon_2 = \langle ?, ? \rangle, \varepsilon_3 = \langle \alpha^{E'_3}, E'_3 \rangle, \varepsilon_4 = \langle \alpha^{E'_4}, E'_4 \rangle$). This case is not possible, since $\langle E_1, \alpha^{E'_1} \rangle \not\leq \langle ?, ? \rangle$.

- Case $[\alpha_3]$ ($\varepsilon_1 = \langle E_1, E'_1 \rangle, \varepsilon_2 = \langle ?, ? \rangle, \varepsilon_3 = \langle E_3, \alpha^{E'_3} \rangle, \varepsilon_4 = \langle E_4, \alpha^{E'_4} \rangle$). This case was discussed in Case $[\alpha_3]$ above.

- ($\varepsilon_1 = \langle \forall X.E_1, \forall X.E'_1 \rangle$). This case is not possible, since $\langle \forall X.E_1, \forall X.E'_1 \rangle \not\leq \langle ?, ? \rangle$.

- ($\varepsilon_i = \langle E_{1i} \rightarrow E_{2i}, E'_{1i} \rightarrow E'_{2i} \rangle$). We have to prove that

$$\langle E_{11} \rightarrow E_{21}, E'_{11} \rightarrow E'_{21} \rangle \circ \langle E_{13} \rightarrow E_{23}, E'_{13} \rightarrow E'_{23} \rangle \leq \langle ?, ? \rangle \circ \langle E_{14} \rightarrow E_{24}, E'_{14} \rightarrow E'_{24} \rangle$$

or what is the same

$$\langle E_{11} \rightarrow E_{21}, E'_{11} \rightarrow E'_{21} \rangle \circ \langle E_{13} \rightarrow E_{23}, E'_{13} \rightarrow E'_{23} \rangle \leq \langle ? \rightarrow ?, ? \rightarrow ? \rangle \circ \langle E_{14} \rightarrow E_{24}, E'_{14} \rightarrow E'_{24} \rangle$$

But, this case was discussed in Case $[\rightarrow]$ above.

- ($\varepsilon_i = \langle E_{1i} \times E_{2i}, E'_{1i} \times E'_{2i} \rangle$). We have to prove that

$$\langle E_{11} \times E_{21}, E'_{11} \times E'_{21} \rangle \circ \langle E_{13} \times E_{23}, E'_{13} \times E'_{23} \rangle \leq \langle ?, ? \rangle \circ \langle E_{14} \times E_{24}, E'_{14} \times E'_{24} \rangle$$

or what is the same:

$$\langle E_{11} \times E_{21}, E'_{11} \times E'_{21} \rangle \circ \langle E_{13} \times E_{23}, E'_{13} \times E'_{23} \rangle \leq \langle ? \times ?, ? \times ? \rangle \circ \langle E_{14} \times E_{24}, E'_{14} \times E'_{24} \rangle$$

This case was discussed in Case $[\times]$.

Case ($[?_3]$ - $\varepsilon_3 = \langle ?, ? \rangle$). Since $\varepsilon_3 \leq \varepsilon_4$, we know that $\varepsilon_4 = \langle ?, ? \rangle$. Therefore, by the transitivity rules, we know that $\varepsilon_1 \circ \varepsilon_3 = \varepsilon_1$ and $\varepsilon_2 \circ \varepsilon_4 = \varepsilon_2$. Thus, we are required to prove that $\varepsilon_1 \leq \varepsilon_2$, but the result follows immediately by premise.

Case ($[?_4]$ - $\varepsilon_4 = \langle ?, ? \rangle$). The proof follows from some of the previous cases.

- ($\varepsilon_1 = \langle ?, ? \rangle$). The results follows immediately, since it was discussed in Case $[?_1]$.
- ($\varepsilon_2 = \langle ?, ? \rangle$). The results follows immediately, since $\varepsilon_1 \circ \varepsilon_3 \leq \langle ?, ? \rangle \circ \langle ?, ? \rangle = \langle ?, ? \rangle$.
- ($\varepsilon_3 = \langle ?, ? \rangle$). The results follows immediately, since it was discussed in Case $[?_3]$.
- ($\varepsilon_i = \langle B, B \rangle$). The results follows immediately, since $\langle B, B \rangle \circ \langle B, B \rangle \leq \langle B, B \rangle \circ \langle ?, ? \rangle$.
- ($\varepsilon_i = \langle X, X \rangle$). This case is not possible, since $\langle X, X \rangle \not\leq \langle ?, ? \rangle$.
- Case $[\alpha_1]$ ($\varepsilon_1 = \langle \alpha^{E_1}, E'_1 \rangle, \varepsilon_2 = \langle \alpha^{E_2}, E'_2 \rangle, \varepsilon_3 = \langle E_3, E'_3 \rangle, \varepsilon_4 = \langle ?, ? \rangle$). This case was discussed in Case $[\alpha_1]$ above.
- Case $[\alpha_2]$ ($\varepsilon_1 = \langle E_1, \alpha^{E'_1} \rangle, \varepsilon_2 = \langle E_2, \alpha^{E'_2} \rangle, \varepsilon_3 = \langle \alpha^{E_3}, E'_3 \rangle, \varepsilon_4 = \langle ?, ? \rangle$). This case is not possible, since $\langle E_3, \alpha^{E'_3} \rangle \not\leq \langle ?, ? \rangle$.
- Case $[\alpha_3]$ ($\varepsilon_1 = \langle E_1, E'_1 \rangle, \varepsilon_2 = \langle E_2, E'_2 \rangle, \varepsilon_3 = \langle E_3, \alpha^{E'_3} \rangle, \varepsilon_4 = \langle ?, ? \rangle$). This case is not possible, since $\langle E_3, \alpha^{E'_3} \rangle \not\leq \langle ?, ? \rangle$.
- ($\varepsilon_1 = \langle \forall X.E_1, \forall X.E'_1 \rangle$). This case is not possible, since $\langle \forall X.E_1, \forall X.E'_1 \rangle \not\leq \langle ?, ? \rangle$.
- ($\varepsilon_i = \langle E_{1i} \rightarrow E_{2i}, E'_{1i} \rightarrow E'_{2i} \rangle$). We have to prove that

$$\langle E_{11} \rightarrow E_{21}, E'_{11} \rightarrow E'_{21} \rangle \circ \langle E_{13} \rightarrow E_{23}, E'_{13} \rightarrow E'_{23} \rangle \leq \langle E_{14} \rightarrow E_{24}, E'_{14} \rightarrow E'_{24} \rangle \circ \langle ?, ? \rangle$$

or what is the same

$$\langle E_{11} \rightarrow E_{21}, E'_{11} \rightarrow E'_{21} \rangle \circ \langle E_{13} \rightarrow E_{23}, E'_{13} \rightarrow E'_{23} \rangle \leq \langle E_{14} \rightarrow E_{24}, E'_{14} \rightarrow E'_{24} \rangle \circ \langle ? \rightarrow ?, ? \rightarrow ? \rangle$$

But, this case was discussed in Case $[\rightarrow]$ above.

- ($\varepsilon_i = \langle E_{1i} \times E_{2i}, E'_{1i} \times E'_{2i} \rangle$). We have to prove that

$$\langle E_{11} \times E_{21}, E'_{11} \times E'_{21} \rangle \mathbin{\dot{\leq}} \langle E_{13} \times E_{23}, E'_{13} \times E'_{23} \rangle \leq \langle E_{14} \times E_{24}, E'_{14} \times E'_{24} \rangle \mathbin{\dot{\leq}} \langle ?, ? \rangle$$

or what is the same:

$$\langle E_{11} \times E_{21}, E'_{11} \times E'_{21} \rangle \mathbin{\dot{\leq}} \langle E_{13} \times E_{23}, E'_{13} \times E'_{23} \rangle \leq \langle E_{14} \times E_{24}, E'_{14} \times E'_{24} \rangle \mathbin{\dot{\leq}} \langle ? \times ?, ? \times ? \rangle$$

This case was discussed in Case $[\times]$.

□

Definition A.48 (Store Precision) $\Xi_1 \leq \Xi_2 \iff \Xi_1 = \Xi'_1, \alpha := G_1, \Xi_2 = \Xi'_2, \alpha := G_2, G_1 \leq G_2$ and $\Xi'_1 \leq \Xi'_2$, or $\Xi_1 = \Xi_2 = \cdot$.

Definition A.49 (Typing Environment Precision) $\Gamma_1 \sqsubseteq \Gamma_2 \iff \Gamma_1 = \Gamma'_1, x : G_1, \Gamma_2 = \Gamma'_2, x : G_2, G_1 \leq G_2$ and $\Gamma'_1 \sqsubseteq \Gamma'_2$, or $\Gamma_1 = \Gamma_2 = \cdot$.

Proposition A.50 (Lift Environment Precision) *If $G_1 \leq G_2$ and $\Xi_1 \leq \Xi_2$, then $\hat{G}_1 \leq \hat{G}_2$, where $\hat{G}_1 = \text{lift}_{\Xi_1}(G_1)$ and $\hat{G}_2 = \text{lift}_{\Xi_2}(G_2)$.*

PROOF. Remember that

$$\text{lift}_{\Xi}(G) = \begin{cases} \text{lift}_{\Xi}(G_1) \rightarrow \text{lift}_{\Xi}(G_2) & G = G_1 \rightarrow G_2 \\ \forall X. \text{lift}_{\Xi}(G_1) & G = \forall X. G_1 \\ \text{lift}_{\Xi}(G_1) \times \text{lift}_{\Xi}(G_2) & G = G_1 \times G_2 \\ \alpha^{\text{lift}_{\Xi}(\Xi(\alpha))} & G = \alpha \\ G & \text{otherwise} \end{cases}$$

The prove follows by the definition of $\hat{G}_1 = \text{lift}_{\Xi_1}(G_1)$ and induction on the structure of the type.

Case ($G_i = B$). The result follows immediately due to $\hat{B} = B \leq B = \hat{B}$.

Case ($G_i = X$). The result follows immediately due to $\hat{X} = X \leq X = \hat{X}$.

Case ($G_i = \alpha$). We are required to prove that $\alpha^{\text{lift}_{\Xi_1}(\Xi_1(\alpha))} \leq \alpha^{\text{lift}_{\Xi_2}(\Xi_2(\alpha))}$, or what is the same $\text{lift}_{\Xi_1}(\Xi_1(\alpha)) \leq \text{lift}_{\Xi_2}(\Xi_2(\alpha))$. Note that $\Xi_1(\alpha) \leq \Xi_2(\alpha)$ due to $\Xi_1 \leq \Xi_2$. The result follows immediately by the induction hypothesis on $\Xi_1(\alpha) \leq \Xi_2(\alpha)$ and $\Xi_1 \leq \Xi_2$.

Case ($G_i = \forall X. G'_i$). We know that $G'_1 \leq G'_2$. We are required to prove that $\forall X. \text{lift}_{\Xi_1}(G'_1) \leq \forall X. \text{lift}_{\Xi_2}(G'_2)$, or what is the same $\text{lift}_{\Xi_1}(G'_1) \leq \text{lift}_{\Xi_2}(G'_2)$. By the induction hypothesis on $G'_1 \leq G'_2$ and $\Xi_1 \leq \Xi_2$ the result follows immediately.

Case ($G_i = G'_i \rightarrow G''_i$). We know that $G'_1 \leq G'_2$ and $G''_1 \leq G''_2$. We are required to prove that $\text{lift}_{\Xi_1}(G'_1) \rightarrow \text{lift}_{\Xi_1}(G''_1) \leq \text{lift}_{\Xi_2}(G'_2) \rightarrow \text{lift}_{\Xi_2}(G''_2)$, or what is the same $\text{lift}_{\Xi_1}(G'_1) \leq \text{lift}_{\Xi_2}(G'_2)$ and $\text{lift}_{\Xi_1}(G''_1) \leq \text{lift}_{\Xi_2}(G''_2)$. By the induction hypothesis on $G'_1 \leq G'_2$ and $G''_1 \leq G''_2$ with $\Xi_1 \leq \Xi_2$ the result follows immediately.

Case ($G_i = G'_i \times G''_i$). This case is similar to the function case above.

Case ($G_1 = ?$). Then $G_2 = ?$. The result follows immediately due to $\hat{?} = ? \leq ? = \hat{?}$.

Case ($G_2 = ?$). Note that $\hat{G}_2 = \hat{?} = ?$. Therefore, we are required to prove that $\hat{G}_1 \leq ?$.

- Case ($G_1 = B$). The result follows immediately, $\hat{B} = B \leq ?$.
- Case ($G_1 = X$). This case is not possible due to $X \not\leq ?$.
- Case ($G_1 = \alpha$). This case is not possible due to $\alpha \not\leq ?$.
- Case ($G_1 = \forall X.G'_1$). This case is not possible due to $\forall X.G'_1 \not\leq ?$.
- Case ($G_1 = G'_1 \rightarrow G'_2$). We are required to prove that $\text{lift}_{\Xi_1}(G'_1) \rightarrow \text{lift}_{\Xi_2}(G'_2) \leq ?$, or what is the same $\text{lift}_{\Xi_1}(G'_1) \rightarrow \text{lift}_{\Xi_2}(G'_2) \leq ? \rightarrow ?$, which follows similar to the function case above.
- Case ($G_1 = G'_1 \times G'_2$). We are required to prove that $\text{lift}_{\Xi_1}(G'_1) \times \text{lift}_{\Xi_2}(G'_2) \leq ?$, or what is the same $\text{lift}_{\Xi_1}(G'_1) \times \text{lift}_{\Xi_2}(G'_2) \leq ? \times ?$, which follows similar to the pair case above.

□

Proposition A.51 (Unlift Evidence Types Preserves Precision) *If $E_1 \leq E_2$ then $\text{unlift}(E_1) \leq \text{unlift}(E_2)$.*

PROOF. Remember that

$$\text{unlift}(E) = \begin{cases} B & E = B \\ \text{unlift}(E_1) \rightarrow \text{unlift}(E_2) & E = E_1 \rightarrow E_2 \\ \forall X.\text{unlift}(E_1) & E = \forall X.E_1 \\ \text{unlift}(E_1) \times \text{unlift}(E_2) & E = E_1 \times E_2 \\ \alpha & E = \alpha^{E_1} \\ X & E = X \\ ? & E = ? \end{cases}$$

The prove follows by the definition of $\text{unlift}(E_1)$ and induction on the structure of the type.

Case ($G_i = B$). The result follows immediately due to $\text{unlift}(B) = B \leq B = \text{unlift}(B)$.

Case ($G_i = X$). The result follows immediately due to $\text{unlift}(X) = X \leq X = \text{unlift}(X)$.

Case ($G_i = \alpha^{E_i}$). The result follows immediately due to $\text{unlift}(\alpha^{E_1}) = \alpha \leq \alpha = \text{unlift}(\alpha^{E_2})$.

Case ($E_i = \forall X.E'_i$). We know that $E'_1 \leq E'_2$. We are required to prove that $\forall X.\text{unlift}(E'_1) \leq \forall X.\text{unlift}(E'_2)$, or what is the same $\text{unlift}(E'_1) \leq \text{unlift}(E'_2)$. By the induction hypothesis on $E'_1 \leq E'_2$ the result follows immediately.

Case ($E_i = E'_i \rightarrow E''_i$). We know that $E'_1 \leq E'_2$ and $E''_1 \leq E''_2$. We are required to prove that $\text{unlift}(E'_1) \rightarrow \text{unlift}(E''_1) \leq \text{unlift}(E'_2) \rightarrow \text{unlift}(E''_2)$, or what is the same $\text{unlift}(E'_1) \leq \text{unlift}(E'_2)$ and $\text{unlift}(E''_1) \leq \text{unlift}(E''_2)$. By the induction hypothesis on $E'_1 \leq E'_2$ and $E''_1 \leq E''_2$ the result follows immediately.

Case ($E_i = E'_i \times E''_i$). This case is similar to the function case above.

Case ($E_1 = ?$). Then $E_2 = ?$. The result follows immediately due to $\text{unlift}(?) = ? \leq ? = \text{unlift}(?)$.

Case ($E_2 = ?$). Note that $\text{unlift}(E_2) = \text{unlift}(?) = ?$. Therefore, we are required to prove that $\text{unlift}(E_1) \leq ?$.

- Case ($E_1 = B$). The result follows immediately, $\text{unlift}(B) = B \leq ?$.

- Case ($E_1 = X$). This case is not possible due to $X \not\leq ?$.
- Case ($E_1 = \alpha$). This case is not possible due to $\alpha \not\leq ?$.
- Case ($E_1 = \forall X.E'_1$). This case is not possible due to $\forall X.E'_1 \not\leq ?$.
- Case ($E_1 = E'_1 \rightarrow E'_2$). We are required to prove that $\text{unlift}(E'_1) \rightarrow \text{unlift}(E'_2) \leq ?$, or what is the same $\text{unlift}(E'_1) \rightarrow \text{unlift}(E'_2) \leq ? \rightarrow ?$, which follows similar to the function case above.
- Case ($E_1 = E'_1 \times E'_2$). We are required to prove that $\text{unlift}(E'_1) \times \text{unlift}(E'_2) \leq ?$, or what is the same $\text{unlift}(E'_1) \times \text{unlift}(E'_2) \leq ? \times ?$, which follows similar to the pair case above.

□

Proposition A.52 *If $\varepsilon_1 \leq \varepsilon_2$, $G_1 \leq G_2$, $\Xi_1 \leq \Xi_2$, $\alpha := G_1 \in \Xi_1$, $\alpha := G_2 \in \Xi_2$ and $\varepsilon_1[\hat{\alpha}_1/X]$ is defined, then*

- $\varepsilon_1[\hat{\alpha}_1/X] \leq \varepsilon_2[\hat{\alpha}_2/X]$.
- $\langle E_1^*[\hat{\alpha}_1/X], E_1^*[\hat{G}_1/X] \rangle \leq \langle E_2^*[\hat{\alpha}_2/X], E_2^*[\hat{G}_2/X] \rangle$.

where $E_1^* = \text{lift}_{\Xi_1}(\text{unlift}(\pi_2(\varepsilon_1)))$, $E_2^* = \text{lift}_{\Xi_2}(\text{unlift}(\pi_2(\varepsilon_2)))$, $\hat{\alpha}_1 = \text{lift}_{\Xi_1}(\alpha_1)$, $\hat{\alpha}_2 = \text{lift}_{\Xi_2}(\alpha_2)$, $\hat{G}_1 = \text{lift}_{\Xi_1}(G_1)$ and $\hat{G}_2 = \text{lift}_{\Xi_2}(G_2)$.

PROOF. Note that $\hat{\alpha}_1 \leq \hat{\alpha}_2$ and $\hat{G}_1 \leq \hat{G}_2$ by Proposition A.50. Suppose that $\varepsilon_1 = \langle E, E' \rangle$ and $\varepsilon_2 = \langle E'', E''' \rangle$. We are required to prove that

$$\begin{aligned} \varepsilon_1[\hat{\alpha}_1/X] &= \langle E[\hat{\alpha}_1/X], E'[\hat{\alpha}_1/X] \rangle \leq \langle E''[\hat{\alpha}_2/X], E'''[\hat{\alpha}_2/X] \rangle = \varepsilon_2[\hat{\alpha}_2/X] \\ \varepsilon_1^* &= \langle E_1^*[\hat{\alpha}_1/X], E_1^*[\hat{G}_1/X] \rangle \leq \langle E_2^*[\hat{\alpha}_2/X], E_2^*[\hat{G}_2/X] \rangle = \varepsilon_2^* \end{aligned}$$

We follow by case analysis on the evidence type, the definition of consistent transitivity for $=$ and the definition of precision.

Case ($\varepsilon_i = \langle B, B \rangle$). The results follows immediately because $\varepsilon_1[\hat{\alpha}_1/X] = \varepsilon_2[\hat{\alpha}_2/X] = \varepsilon_1^* = \varepsilon_2^* = \langle B, B \rangle$.

Case ($\varepsilon_i = \langle X, X \rangle$). We are required to prove that $\varepsilon_1[\hat{\alpha}_1/X] = \langle \hat{\alpha}_1, \hat{\alpha}_1 \rangle \leq \langle \hat{\alpha}_2, \hat{\alpha}_2 \rangle = \varepsilon_2[\hat{\alpha}_2/X]$, which follows immediately due to $\hat{\alpha}_1 \leq \hat{\alpha}_2$. Also, we are required to prove that $\varepsilon_1^* = \langle \hat{\alpha}_1, \hat{G}_1 \rangle \leq \langle \hat{\alpha}_2, \hat{G}_2 \rangle = \varepsilon_2^*$, which follows immediately due to $\hat{\alpha}_1 \leq \hat{\alpha}_2$ and $\hat{G}_1 \leq \hat{G}_2$.

Case ($\varepsilon_i = \langle Y, Y \rangle$). The results follows immediately because $\varepsilon_1[\hat{\alpha}_1/X] = \varepsilon_2[\hat{\alpha}_2/X] = \varepsilon_1^* = \varepsilon_2^* = \langle Y, Y \rangle$.

Case ($\varepsilon_i = \langle \beta^{E_i}, E'_i \rangle$). The results follows immediately because $\varepsilon_1[\hat{\alpha}_1/X] = \langle \beta^{E_1}, E'_1 \rangle \leq \langle \beta^{E_2}, E'_2 \rangle = \varepsilon_2[\hat{\alpha}_2/X]$ by premise (note that X can not be free in $\langle \beta^{E_i}, E'_i \rangle$). Also, we are required to prove that $\varepsilon_1^* \leq \varepsilon_2^*$, but the result follows immediately by Proposition A.51 and Proposition A.50.

Case ($\varepsilon_i = \langle E_i, \beta^{E_i} \rangle$). Similar to the previous case.

Case ($\varepsilon_i = \langle \forall Y.E_i, \forall Y.E'_i \rangle$). By the definition of \leq , we know that $\langle E_1, E'_1 \rangle \leq \langle E_2, E'_2 \rangle$. We are required to prove that

$$\varepsilon_1[\hat{\alpha}_1/X] = \langle \forall Y.E_1[\hat{\alpha}_1/X], \forall Y.E'_1[\hat{\alpha}_1/X] \rangle \leq \langle \forall Y.E_2[\hat{\alpha}_2/X], \forall Y.E'_2[\hat{\alpha}_2/X] \rangle = \varepsilon_2[\hat{\alpha}_2/X]$$

or what is the same

$$\langle E_1, E'_1 \rangle[\hat{\alpha}_1/X] = \langle E_1[\hat{\alpha}_1/X], E'_1[\hat{\alpha}_1/X] \rangle \leq \langle E_2[\hat{\alpha}_2/X], E'_2[\hat{\alpha}_2/X] \rangle = \langle E_2, E'_2 \rangle[\hat{\alpha}_2/X]$$

By the induction hypothesis on $\langle E_1, E'_1 \rangle \leq \langle E_2, E'_2 \rangle$ the result follows immediately.

Also we are required to prove

$$\varepsilon_1^* = \langle E_1^*[\hat{\alpha}_1/X], E_1^*[\hat{G}_1/X] \rangle \leq \langle E_2^*[\hat{\alpha}_2/X], E_2^*[\hat{G}_2/X] \rangle = \varepsilon_2^*$$

Note that $E_1^* = \text{lift}_{\Xi_1}(\text{unlift}(\forall Y.E'_1)) = \forall Y.\text{lift}_{\Xi_1}(\text{unlift}(E'_1)) = \forall Y.E_{11}^*$ and $E_2^* = \text{lift}_{\Xi_2}(\text{unlift}(\forall Y.E'_2)) = \forall Y.\text{lift}_{\Xi_2}(\text{unlift}(E'_2)) = \forall Y.E_{22}^*$. Therefore, we are required to prove

$$\langle E_{11}^*[\hat{\alpha}_1/X], E_{11}^*[\hat{G}_1/X] \rangle \leq \langle E_{22}^*[\hat{\alpha}_2/X], E_{22}^*[\hat{G}_2/X] \rangle$$

By the induction hypothesis on $\langle E_1, E'_1 \rangle \leq \langle E_2, E'_2 \rangle$ the result follows immediately.

Case ($\varepsilon_i = \langle E_{1i} \rightarrow E_{2i}, E'_{1i} \rightarrow E'_{2i} \rangle$). By the definition of \leq , we know that $\langle E_{11}, E'_{11} \rangle \leq \langle E_{12}, E'_{12} \rangle$ and $\langle E_{21}, E'_{21} \rangle \leq \langle E_{22}, E'_{22} \rangle$. We are required to prove that

$$\begin{aligned} \varepsilon_1[\hat{\alpha}_1/X] &= \langle E_{11}[\hat{\alpha}_1/X] \rightarrow E_{12}[\hat{\alpha}_1/X], E'_{11}[\hat{\alpha}_1/X] \rightarrow E'_{12}[\hat{\alpha}_1/X] \rangle \leq \\ &\langle E_{12}[\hat{\alpha}_2/X] \rightarrow E_{21}[\hat{\alpha}_2/X], E'_{12}[\hat{\alpha}_2/X] \rightarrow E'_{21}[\hat{\alpha}_2/X] \rangle = \varepsilon_2[\hat{\alpha}_2/X] \end{aligned}$$

or what is the same

$$\langle E_{11}[\hat{\alpha}_1/X], E'_{11}[\hat{\alpha}_1/X] \rangle \leq \langle E_{12}[\hat{\alpha}_2/X], E'_{12}[\hat{\alpha}_2/X] \rangle$$

and

$$\langle E_{12}[\hat{\alpha}_1/X], E'_{12}[\hat{\alpha}_1/X] \rangle \leq \langle E_{21}[\hat{\alpha}_2/X], E'_{21}[\hat{\alpha}_2/X] \rangle$$

By the induction hypothesis on $\langle E_{11}, E'_{11} \rangle \leq \langle E_{12}, E'_{12} \rangle$ and $\langle E_{21}, E'_{21} \rangle \leq \langle E_{22}, E'_{22} \rangle$ the result follows immediately.

Also we are required to prove

$$\varepsilon_1^* = \langle E_1^*[\hat{\alpha}_1/X], E_1^*[\hat{G}_1/X] \rangle \leq \langle E_2^*[\hat{\alpha}_2/X], E_2^*[\hat{G}_2/X] \rangle = \varepsilon_2^*$$

Note that $E_1^* = \text{lift}_{\Xi_1}(\text{unlift}(E'_{11} \rightarrow E'_{12})) = \text{lift}_{\Xi_2}(\text{unlift}(E'_{11})) \rightarrow \text{lift}_{\Xi_2}(\text{unlift}(E'_{12})) = E_{11}^* \rightarrow E_{12}^*$ and $E_2^* = \text{lift}_{\Xi_2}(\text{unlift}(E'_{21} \rightarrow E'_{22})) = \text{lift}_{\Xi_2}(\text{unlift}(E'_{21})) \rightarrow \text{lift}_{\Xi_2}(\text{unlift}(E'_{22})) = E_{21}^* \rightarrow E_{22}^*$. Therefore, we are required to prove

$$\langle E_{11}^*[\hat{\alpha}_1/X], E_{11}^*[\hat{G}_1/X] \rangle \leq \langle E_{21}^*[\hat{\alpha}_2/X], E_{21}^*[\hat{G}_2/X] \rangle$$

and

$$\langle E_{12}^*[\hat{\alpha}_1/X], E_{12}^*[\hat{G}_1/X] \rangle \leq \langle E_{22}^*[\hat{\alpha}_2/X], E_{22}^*[\hat{G}_2/X] \rangle$$

By the induction hypothesis on $\langle E_{11}, E'_{11} \rangle \leq \langle E_{12}, E'_{12} \rangle$ and $\langle E_{21}, E'_{21} \rangle \leq \langle E_{22}, E'_{22} \rangle$ the result follows immediately.

Case $(\varepsilon_i = \langle E_{1i} \times E_{2i}, E'_{1i} \times E'_{2i} \rangle)$. Similar to the function case.

Case $(\varepsilon_1 = \langle ?, ? \rangle)$. Note that if $\varepsilon_1 = \langle ?, ? \rangle$ then $\varepsilon_2 = \langle ?, ? \rangle$. Therefore, the result follows immediately because $\varepsilon_1[\hat{\alpha}_1] = \varepsilon_2[\hat{\alpha}_2] = \varepsilon_1^* = \varepsilon_2^* = \langle ?, ? \rangle$. This case is trivial,

Case $(\varepsilon_2 = \langle ?, ? \rangle)$. Note that $\varepsilon_2[\hat{\alpha}_2] = \varepsilon_2^* = \langle ?, ? \rangle$. Therefore, we are required to prove that $\varepsilon_1[\hat{\alpha}_1] \leq \langle ?, ? \rangle$ and $\varepsilon_1^* \leq \langle ?, ? \rangle$.

- Case $(\varepsilon_1 = \langle B, B \rangle)$. The result follows immediately, $\varepsilon_1[\hat{\alpha}_1/X] = \varepsilon_1^* = \langle B, B \rangle \leq \langle ?, ? \rangle$.
- Case $(\varepsilon_1 = \langle X, X \rangle)$. This case is not possible due to $\langle X, X \rangle \not\leq \langle ?, ? \rangle$.
- Case $(\varepsilon_1 = \langle \alpha^{E_1}, E'_1 \rangle)$. This case is not possible due to $\langle \alpha^{E_1}, E'_1 \rangle \not\leq \langle ?, ? \rangle$.
- Case $(\varepsilon_1 = \langle E_1, \alpha^{E'_1} \rangle)$. This case is not possible due to $\langle E_1, \alpha^{E'_1} \rangle \not\leq \langle ?, ? \rangle$.
- Case $(\varepsilon_1 = \langle \forall Y.E_1, \forall Y.E'_1 \rangle)$. This case is not possible due to $\langle \forall Y.E_1, \forall Y.E'_1 \rangle \not\leq \langle ?, ? \rangle$.
- Case $(\varepsilon_1 = \langle E_{11} \rightarrow E_{12}, E'_{11} \rightarrow E'_{12} \rangle)$. We are required to prove that $\varepsilon_1[\hat{\alpha}_1] \leq \langle ?, ? \rangle$ and $\varepsilon_1^* \leq \langle ?, ? \rangle$, or what is the same $\varepsilon_1[\hat{\alpha}_1] \leq \langle ? \rightarrow ?, ? \rightarrow ? \rangle$ and $\varepsilon_1^* \leq \langle ? \rightarrow ?, ? \rightarrow ? \rangle$, which follows similar to the function case above.
- Case $(\varepsilon_1 = \langle E_{11} \times E_{12}, E'_{11} \times E'_{12} \rangle)$. We are required to prove that $\varepsilon_1[\hat{\alpha}_1] \leq \langle ?, ? \rangle$ and $\varepsilon_1^* \leq \langle ?, ? \rangle$, or what is the same $\varepsilon_1[\hat{\alpha}_1] \leq \langle ? \times ?, ? \times ? \rangle$ and $\varepsilon_1^* \leq \langle ? \times ?, ? \times ? \rangle$, which follows similar to the pair case above.

□

Proposition A.53 *If $\varepsilon_1 \sqsubseteq \varepsilon_2$, $G_1 \leq G_2$, $\Xi_1 \leq \Xi_2$, $\alpha := G_1 \in \Xi_1$, $\alpha := G_2 \in \Xi_2$ and $\varepsilon_1[\hat{\alpha}_1/X]$ is defined, then $\varepsilon_1[\hat{\alpha}_1/X] \sqsubseteq \varepsilon_2[\hat{\alpha}_2/X]$, where $\hat{\alpha}_1 = \text{lift}_{\Xi_1}(\alpha)$ and $\hat{\alpha}_2 = \text{lift}_{\Xi_2}(\alpha)$.*

PROOF. Similar to Proposition A.52. □

Proposition A.54 (Monotonicity of Evidence Instantiation) *If $\varepsilon_1 \leq \varepsilon_2$, $G_1 \leq G_2$, $\Xi_1 \leq \Xi_2$, $\alpha := G_1 \in \Xi_1$, $\alpha := G_2 \in \Xi_2$ and $\varepsilon_1[\hat{\alpha}_1]$ is defined, then*

- $\hat{\alpha}_1 \leq \hat{\alpha}_2$.
- $\varepsilon_1[\hat{\alpha}_1] \leq \varepsilon_2[\hat{\alpha}_2]$.
- $\varepsilon_{1out} \leq \varepsilon_{2out}$.

where $\hat{\alpha}_1 = \text{lift}_{\Xi_1}(\alpha)$ and $\hat{\alpha}_2 = \text{lift}_{\Xi_2}(\alpha)$.

PROOF. This result $\hat{\alpha}_1 \leq \hat{\alpha}_2$ follows immediately by the Proposition A.50.

Remember that

$$\varepsilon_{out} \triangleq \langle E_*[\alpha^E], E_*[E'] \rangle \quad \text{where } E_* = \text{lift}_{\Xi}(\text{unlift}(\pi_2(\varepsilon))), \alpha^E = \text{lift}_{\Xi'}(\alpha), E' = \text{lift}_{\Xi}(G')$$

Note that $\varepsilon_1[\hat{\alpha}_1]$ only succeed if $\varepsilon_1 = \langle \forall X.E, \forall X.E' \rangle$. Since $\varepsilon_1 \leq \varepsilon_2$ and $\varepsilon_1 = \langle \forall X.E, \forall X.E' \rangle$, then $\varepsilon_2 = \langle \forall X.E'', \forall X.E''' \rangle$. Let suppose that $\varepsilon'_1 = \langle E, E' \rangle$ and $\varepsilon'_2 = \langle E'', E''' \rangle$. Then we are required to prove that

$$\begin{aligned}\varepsilon_1[\hat{\alpha}_1] &= \varepsilon'_1[\hat{\alpha}_1/X] = \langle E[\hat{\alpha}_1/X], E'[\hat{\alpha}_1/X] \rangle \leq \langle E''[\hat{\alpha}_2/X], E'''[\hat{\alpha}_2/X] \rangle = \varepsilon'_2[\hat{\alpha}_2/X] = \varepsilon_2[\hat{\alpha}_2] \\ \varepsilon_{1out} &= \langle E_1^*[\hat{\alpha}_1/X], E_1^*[\hat{G}_1/X] \rangle \leq \langle E_2^*[\hat{\alpha}_2/X], E_2^*[\hat{G}_2/X] \rangle = \varepsilon_{2out}\end{aligned}$$

where $E_1^* = \text{lift}_{\Xi_1}(\text{unlift}(E'))$, $E_2^* = \text{lift}_{\Xi_2}(\text{unlift}(E'''))$, $\hat{G}_1 = \text{lift}_{\Xi_1}(G_1)$ and $\hat{G}_2 = \text{lift}_{\Xi_2}(G_2)$.

By the Proposition A.52 the result follows immediately. \square

Proposition A.55 *If $G_1^* \sqsubseteq G_2^*$ and $G'_1 \sqsubseteq G'_2$ then $G_1^*[G'_1/X] \sqsubseteq G_2^*[G'_2/X]$.*

PROOF. Follow by induction on $G_1^* \sqsubseteq G_2^*$.

Case ($B \sqsubseteq B$). The results follows immediately due to $B[G'_1/X] = B \sqsubseteq B = B[G'_2/X]$.

Case ($Y \sqsubseteq Y$). If $Y = X$, the results follows immediately due to $X[G'_1/X] = G'_1 \sqsubseteq G'_2 = X[G'_2/X]$ and $G'_1 \sqsubseteq G'_2$ by premise. If $Y \not\rightarrow X$, the results, also, follows immediately due to $Y[G'_1/X] = Y \sqsubseteq Y = Y[G'_2/X]$.

Case ($\alpha \sqsubseteq \alpha$). The results follows immediately due to $\alpha[G'_1/X] = \alpha \sqsubseteq \alpha = \alpha[G'_2/X]$.

Case ($G \sqsubseteq ?$). The results follows immediately due to $G[G'_1/X] \sqsubseteq ? = ?[G'_2/X]$.

Case ($\forall X.G_1 \sqsubseteq \forall X.G_2$). We know that

$$\frac{G_1 \sqsubseteq G_2}{\forall X.G_1 \sqsubseteq \forall X.G_2}$$

By the definition of \sqsubseteq , we know that $G_1 \sqsubseteq G_2$. We are required to prove that

$$(\forall X.G_1)[G'_1/X] = (\forall X.G_1[G'_1/X]) \sqsubseteq (\forall X.G_2[G'_2/X]) = (\forall X.G_2)[G'_2/X]$$

Or what is the same that $(G_1[G'_1/X]) \sqsubseteq (G_2[G'_2/X])$. But the result follows immediately by the induction hypothesis on $G_1 \sqsubseteq G_2$.

Case ($G_1 \rightarrow G_2 \sqsubseteq G_3 \rightarrow G_4$). We know that

$$\frac{G_1 \sqsubseteq G_3 \quad G_2 \sqsubseteq G_4}{G_1 \rightarrow G_2 \sqsubseteq G_3 \rightarrow G_4}$$

By the definition of \sqsubseteq , we know that $G_1 \sqsubseteq G_3$ and $G_2 \sqsubseteq G_4$. We are required to prove that

$$(G_1 \rightarrow G_2)[G'_1/X] = (G_1[G'_1/X] \rightarrow G_2[G'_1/X]) \sqsubseteq (G_3[G'_2/X] \rightarrow G_4[G'_2/X]) = (G_3 \rightarrow G_4)[G'_2/X]$$

Or what is the same that $G_1[G'_1/X] \sqsubseteq G_3[G'_2/X]$ and $G_2[G'_1/X] \sqsubseteq G_4[G'_2/X]$. But the result follows immediately by the induction hypothesis on $G_1 \sqsubseteq G_3$ and $G_2 \sqsubseteq G_4$.

Case ($G_1 \times G_2 \sqsubseteq G_3 \times G_4$). We know that

$$\frac{G_1 \sqsubseteq G_3 \quad G_2 \sqsubseteq G_4}{G_1 \times G_2 \sqsubseteq G_3 \times G_4}$$

By the definition of \sqsubseteq , we know that $G_1 \sqsubseteq G_3$ and $G_2 \sqsubseteq G_4$. We are required to prove that

$$(G_1 \times G_2)[G'_1/X] = (G_1[G'_1/X] \times G_2[G'_1/X]) \sqsubseteq (G_3[G'_2/X] \times G_4[G'_2/X]) = (G_3 \times G_4)[G'_2/X]$$

Or what is the same that $G_1[G'_1/X] \sqsubseteq G_3[G'_2/X]$ and $G_2[G'_1/X] \sqsubseteq G_4[G'_2/X]$. But the result follows immediately by the induction hypothesis on $G_1 \sqsubseteq G_3$ and $G_2 \sqsubseteq G_4$.

□

Proposition A.56 *If $G_1 \sqsubseteq G_2$ and $G'_1 \leq G'_2$ then $G_1[G'_1/X] \sqsubseteq G_2[G'_2/X]$.*

PROOF. By Proposition A.61 and Proposition A.55 the results follows immediately. □

Proposition A.57 *If $G_1 \leq G_2$ and $G'_1 \leq G'_2$ then $G_1[G'_1/X] \leq G_2[G'_2/X]$.*

PROOF. Straightforward induction on $G_1 \leq G_2$. Very similar to Proposition A.55. □

Proposition A.58 *If $G_1 \rightarrow G_2$ then $G_1[\alpha/X] \rightarrow G_2[\alpha/X]$.*

PROOF. By induction on the definition of $G_1 \rightarrow G_2$. □

A.5.3 Weak Dynamic Gradual Guarantee for GSF

In this section, we present the proof of the weak dynamic gradual guarantee for $\text{GSF}\varepsilon$ previously presented and the auxiliary Propositions and Definitions.

Proposition A.59 (Monotonicity of Evidence Substitution) *If $\Omega \vdash \Xi_1 \triangleright s_1^* : G_1^* \leq \Xi_2 \triangleright s_2^* : G_2^*$ and $\Xi_1 \leq \Xi_2$, then $\Omega[\alpha/X] \vdash \Xi_1 \triangleright s_1^*[\hat{\alpha}_1/X] : G_1^*[\alpha/X] \leq \Xi_2 \triangleright s_2^*[\hat{\alpha}_2/X] : G_2^*[\alpha/X]$, where $\alpha := G_1^{**} \in \Xi_1$, $\alpha := G_2^{**} \in \Xi_2$, $\hat{\alpha}_1 = \text{lift}_{\Xi_1}(\alpha)$ and $\hat{\alpha}_2 = \text{lift}_{\Xi_2}(\alpha)$.*

PROOF. We follow by induction on $\Omega \vdash \Xi_1 \triangleright s_1^* : G_1^* \leq \Xi_2 \triangleright s_2^* : G_2^*$. We avoid the notation $\Omega \vdash \Xi_1 \triangleright s_1^* : G_1^*[\alpha/X] \leq \Xi_2 \triangleright s_2^* : G_2^*[\alpha/X]$, and use $s_1^* \leq s_2^*$ instead, for simplicity, when the typing environments are not relevant.

Case ($b \leq b$). The results follows immediately due to $b[\hat{\alpha}_1/X] = b \leq b = b[\hat{\alpha}_2/X]$.

Case ($x \leq x$). The results follows immediately due to $x[\hat{\alpha}_1/X] = x \leq x = x[\hat{\alpha}_2/X]$.

Case ($(\lambda x : G_1.t_1) \leq (\lambda x : G_2.t_2)$). We know that

$$\frac{\Omega \vdash \Xi_1 \triangleright t_1 : G'_1 \leq \Xi_2 \triangleright t_2 : G'_2 \quad G_1 \sqsubseteq G_2}{(\lambda x : G_1.t_1) \leq (\lambda x : G_2.t_2)}$$

We are required to show

$$(\lambda x : G_1.t_1)[\hat{\alpha}_1/X] = (\lambda x : G_1[\alpha/X].t_1[\hat{\alpha}_1/X]) \leq (\lambda x : G_2[\alpha/X].t_2[\hat{\alpha}_2/X]) = (\lambda x : G_2.t_2)[\hat{\alpha}_2/X]$$

Note that $G_1[\alpha/X] \sqsubseteq G_2[\alpha/X]$, by Proposition A.56.

Therefore, we are required to prove

$$\Omega, x : G_1[\alpha/X] \sqsubseteq G_2[\alpha/X] \vdash \Xi_1 \triangleright (t_1[\hat{\alpha}_1/X]) : G'_1[\alpha/X] \leq \Xi_2 \triangleright (t_2[\hat{\alpha}_2/X]) : G'_2[\alpha/X]$$

But the results follows immediately by the induction hypothesis on

$$\Omega, x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright t_1 : G'_1 \leq \Xi_2 \triangleright t_2 : G'_2$$

Case $((\Lambda Y.t_1) \leq (\Lambda Y.t_2))$. We know that

$$\frac{t_1 \leq t_2}{(\Lambda Y.t_1) \leq (\Lambda Y.t_2)}$$

We are required to show

$$(\Lambda Y.t_1)[\hat{\alpha}_1/X] = (\Lambda Y.t_1[\hat{\alpha}_1/X]) \leq (\Lambda Y.t_2[\hat{\alpha}_2/X]) = (\Lambda Y.t_2)[\hat{\alpha}_2/X]$$

Therefore, we are required to prove $(t_1[\hat{\alpha}_1/X]) \leq (t_2[\hat{\alpha}_2/X])$. But the results follows immediately by the induction hypothesis on $t_1 \leq t_2$.

Case $(t_1 t_2 \leq t_1 t'_2)$. We know that

$$\frac{t_1 \leq t'_1 \quad t_2 \leq t'_2}{t_1 t_2 \leq t_1 t'_2}$$

We are required to show

$$(t_1 t_2)[\hat{\alpha}_1/X] = t_1[\hat{\alpha}_1/X] t_2[\hat{\alpha}_1/X] \leq (t'_1[\hat{\alpha}_2/X] t'_2[\hat{\alpha}_2/X]) = (t'_1 t'_2)[\hat{\alpha}_2/X]$$

Therefore, we are required to prove $t_1[\hat{\alpha}_1/X] \leq t'_1[\hat{\alpha}_2/X]$ and $t_2[\hat{\alpha}_1/X] \leq t'_2[\hat{\alpha}_2/X]$. But the results follows immediately by the induction hypothesis on $t_1 \leq t'_1$ and $t_2 \leq t'_2$.

Case $(t_1 [G_1] \leq t_2 [G_2])$. We know that

$$\frac{t_1 \leq t_2 \quad G_1 \leq G_2}{t_1 [G_1] \leq t_2 [G_2]}$$

We are required to show

$$(t_1 [G_1])[\hat{\alpha}_1/X] = (t_1[\hat{\alpha}_1/X] [G_1[\alpha/X]]) \leq (t_2[\hat{\alpha}_2/X] [G_2[\alpha/X]]) = (t_2 [G_2])[\hat{\alpha}_2/X]$$

Note that $G_1[\alpha/X] \leq G_2[\alpha/X]$ by Proposition A.57 and $G_1 \leq G_2$.

Therefore, we are required to prove $(t_1[\hat{\alpha}_1/X]) \leq (t_2[\hat{\alpha}_2/X])$. But the results follows immediately by the induction hypothesis on $t_1 \leq t_2$.

Case $(\varepsilon_1 s_1 :: G_1 \leq \varepsilon_2 s_2 :: G_2)$.

$$\frac{\varepsilon_1 \leq \varepsilon_2 \quad s_1 \leq s_2 \quad G_1 \sqsubseteq G_2}{\varepsilon_1 s_1 :: G_1 \leq \varepsilon_2 s_2 :: G_2}$$

We are required to show

$$(\varepsilon_1 s_1 :: G_1)[\hat{\alpha}_1/X] = (\varepsilon_1[\hat{\alpha}_1/X] s_1[\hat{\alpha}_1/X] :: G_1[\alpha/X]) \leq (\varepsilon_2[\hat{\alpha}_2/X] s_2[\hat{\alpha}_2/X] :: G_2[\alpha/X]) = (\varepsilon_2 s_2 :: G_2)[\hat{\alpha}_2/X]$$

Note that by Proposition A.52 and $\varepsilon_1 \leq \varepsilon_2$, we know that $\varepsilon_1[\hat{\alpha}_1/X] \leq \varepsilon_2[\hat{\alpha}_2/X]$. Also, by Proposition A.56 and $G_1 \sqsubseteq G_2$, we know that $G_1[\alpha/X] \sqsubseteq G_2[\alpha/X]$.

Therefore, we are required to prove $(s_1[\hat{\alpha}_1/X]) \leq (s_2[\hat{\alpha}_2/X])$. But the results follows immediately by the induction hypothesis on $s_1 \leq s_2$.

Case $(\varepsilon_{G_1} t'_1 :: G_1 \leq \varepsilon_{G_2} t'_2 :: G_2)$.

$$\frac{\Omega \vdash \Xi_1 \triangleright t'_1 : G'_1 \leq \Xi_2 \triangleright t'_2 : G'_2 \quad G_1 \sqsubseteq G_2 \quad G'_1 \rightarrow G_1 \quad G'_2 \rightarrow G_2}{\Omega \vdash \Xi_1 \triangleright \varepsilon_{G_1}(t'_1 :: G_1) : G_1 \leq \Xi_2 \triangleright \varepsilon_{G_2}(t'_2 :: G_2) : G_2}$$

We are required to show

$$(\varepsilon_{G_1} t'_1 :: G_1)[\hat{\alpha}_1/X] = (\varepsilon_{G_1}[\hat{\alpha}_1/X] t'_1[\hat{\alpha}_1/X] :: G_1[\alpha/X]) \leq$$

$$(\varepsilon_{G_2}[\hat{\alpha}_2/X]t'_2[\hat{\alpha}_2/X] :: G_2[\alpha/X]) = (\varepsilon_{G_2}t'_2 :: G_2)[\hat{\alpha}_2/X]$$

Note that since $G_1 \sqsubseteq G_2$ and Proposition A.78, we know that $\varepsilon_{G_1} \sqsubseteq \varepsilon_{G_2}$. Note that by Proposition A.53 and $\varepsilon_{G_1} \sqsubseteq \varepsilon_{G_2}$, we know that $\varepsilon_{G_1}[\hat{\alpha}_1/X] \sqsubseteq \varepsilon_{G_2}[\hat{\alpha}_2/X]$. Also, by Proposition A.56 and $G_1 \sqsubseteq G_2$, we know that $G_1[\alpha/X] \sqsubseteq G_2[\alpha/X]$. By Proposition A.58, we know that $G'_1[\alpha/X] \rightarrow G_1[\alpha/X]$ and $G'_2[\alpha/X] \rightarrow G_2[\alpha/X]$. Therefore, we are required to prove $(t_1[\hat{\alpha}_1/X]) \leq (t_2[\hat{\alpha}_2/X])$. But the results follows immediately by the induction hypothesis on $t_1 \leq t_2$.

□

Proposition A.60 (Substitution Preserves Precision) *If $\Omega', x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright s_1 : G'_1 \leq \Xi_2 \triangleright s_2 : G'_2$ and $\Omega' \vdash \Xi_1 \triangleright v_1 : G_1 \leq \Xi_2 \triangleright v_2 : G_2$, then $\Omega' \vdash \Xi_1 \triangleright s_1[v_1/x] : G'_1 \leq \Xi_2 \triangleright s_2[v_2/x] : G'_2$.*

PROOF. We follow by induction on $\Omega', x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright t_1 : G'_1 \leq \Xi_2 \triangleright t_2 : G'_2$. We avoid the notation $\Omega', x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright t_1 : G'_1 \leq \Xi_2 \triangleright t_2 : G'_2$, and use $t_1 \leq t_2$ instead, for simplicity, when the typing environments are not relevant. Let suppose that $\Omega = \Omega', x : G_1 \sqsubseteq G_2$.

Case ($b \leq b$). The result follows immediately.

Case ($x \leq x$). We know that

$$(\leq_{x_\varepsilon}) \frac{x : G_1 \sqsubseteq G_2 \in \Omega}{\Omega \vdash \Xi_1 \triangleright x : G_1 \leq \Xi_2 \triangleright x : G_2}$$

The result follows immediately due to $\Omega \vdash \Xi_1 \triangleright v_1 : G_1 \leq \Xi_2 \triangleright v_2 : G_2$ and

$$t_1[v_1/x] = x[v_1/x] = v_1 \leq v_2 = x[v_2/x] = t_2[v_2/x]$$

Case ($(\lambda y : G''_1.t'_1) \leq (\lambda y : G''_2.t'_2)$). We know that

$$\frac{\Omega, y : G''_1 \sqsubseteq G''_2 \vdash \Xi_1 \triangleright t'_1 : G'''_1 \leq \Xi_2 \triangleright t'_2 : G'''_2 \quad G''_1 \sqsubseteq G''_2}{\Omega \vdash \Xi_1 \triangleright (\lambda y : G''_1.t'_1) : G'''_1 \rightarrow G'''_1 \leq \Xi_2 \triangleright (\lambda y : G''_2.t'_2) : G'''_2 \rightarrow G'''_2}$$

Note that we are required to prove that $\Omega \vdash \Xi_1 \triangleright (\lambda y : G''_1.t'_1) : G'''_1 \rightarrow G'''_1 \leq \Xi_2 \triangleright (\lambda y : G''_2.t'_2) : G'''_2 \rightarrow G'''_2$.

$$(\lambda y : G''_1.t'_1)[v_1/x] = (\lambda y : G''_1.t'_1[v_1/x]) \leq (\lambda y : G''_2.t'_2[v_2/x]) = (\lambda y : G''_2.t'_2)[v_2/x]$$

or what is the same $\Omega, y : G''_1 \sqsubseteq G''_2 \vdash \Xi_1 \triangleright t'_1[v_1/x] : G'''_1 \leq \Xi_2 \triangleright t'_2[v_2/x] : G'''_2$. But the result follows immediately by the induction hypothesis on $\Omega, y : G''_1 \sqsubseteq G''_2 \vdash \Xi_1 \triangleright t'_1 : G'''_1 \leq \Xi_2 \triangleright t'_2 : G'''_2$.

Case ($(\Lambda X.t'_1) \leq (\Lambda X.t'_2)$). We know that

$$\frac{\Omega \vdash \Xi_1 \triangleright t'_1 : G''_1 \leq \Xi_2 \triangleright t'_2 : G''_2}{\Omega \vdash \Xi_1 \triangleright (\Lambda X.t'_1) : \forall X.G''_1 \leq \Xi_2 \triangleright (\Lambda X.t'_2) : \forall X.G''_2}$$

Note that we are required to prove that $\Omega \vdash \Xi_1 \triangleright (\Lambda X.t'_1) : \forall X.G''_1 \leq \Xi_2 \triangleright (\Lambda X.t'_2) : \forall X.G''_2$.

$$(\Lambda X.t'_1)[v_1/x] = (\Lambda X.t'_1[v_1/x]) \leq (\Lambda X.t'_2[v_2/x]) = (\Lambda X.t'_2)[v_2/x]$$

or what is the same $\Omega \vdash \Xi_1 \triangleright t'_1[v_1/x] : G''_1 \leq \Xi_2 \triangleright t'_2[v_2/x] : G''_2$. But the result follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t'_1 : G''_1 \leq \Xi_2 \triangleright t'_2 : G''_2$.

Case $(t'_1 t'_2 \leq t''_1 t''_2)$. We know that

$$\frac{\Omega \vdash \Xi_1 \triangleright t'_1 : G''_1 \rightarrow G'''_1 \leq \Xi_2 \triangleright t'_2 : G''_2 \rightarrow G'''_2 \quad \Omega \vdash \Xi_1 \triangleright t''_1 : G''_1 \leq \Xi_2 \triangleright t''_2 : G''_2}{\Omega \vdash \Xi_1 \triangleright t'_1 t'_1 : G'''_1 \leq \Xi_2 \triangleright t'_2 t'_2 : G'''_2}$$

Note that we are required to prove that $\Omega \vdash \Xi_1 \triangleright t'_1 t'_1 : G'''_1 \leq \Xi_2 \triangleright t'_2 t'_2 : G'''_2$.

$$(t'_1 t'_1)[v_1/x] = t'_1[v_1/x] t'_1[v_1/x] \leq t'_2[v_2/x] t'_2[v_2/x] = (t'_2 t'_2)[v_2/x]$$

or what is the same $\Omega \vdash \Xi_1 \triangleright t'_1[v_1/x] : G''_1 \rightarrow G'''_1 \leq \Xi_2 \triangleright t'_2[v_2/x] : G''_2 \rightarrow G'''_2$ and $\Omega \vdash \Xi_1 \triangleright t''_1[v_1/x] : G''_1 \leq \Xi_2 \triangleright t''_2[v_2/x] : G''_2$. But the result follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t'_1 : G''_1 \rightarrow G'''_1 \leq \Xi_2 \triangleright t'_2 : G''_2 \rightarrow G'''_2$ and $\Omega \vdash \Xi_1 \triangleright t''_1 : G''_1 \leq \Xi_2 \triangleright t''_2 : G''_2$.

Case $(t'_1 [G''_1] \leq t'_2 [G''_2])$.

$$\frac{\Omega \vdash \Xi_1 \triangleright t'_1 : \forall X. G'''_1 \leq \Xi_2 \triangleright t'_2 : \forall X. G'''_2 \quad G''_1 \leq G''_2}{\Omega \vdash \Xi_1 \triangleright t'_1 [G''_1] : G'''_1[G''_1/X] \leq \Xi_2 \triangleright t'_2 [G''_2] : G'''_2[G''_2/X]}$$

Note that we are required to prove that $\Omega \vdash \Xi_1 \triangleright t'_1 [G''_1] : G'''_1[G''_1/X] \leq \Xi_2 \triangleright t'_2 [G''_2] : G'''_2[G''_2/X]$.

$$(t'_1 [G''_1])[v_1/x] = (t'_1[v_1/x] [G''_1]) \leq (t'_2[v_2/x] [G''_2]) = (t'_2 [G''_2])[v_2/x]$$

or what is the same $\Omega \vdash \Xi_1 \triangleright t'_1[v_1/x] : G'''_1[G''_1/X] \leq \Xi_2 \triangleright t'_2[v_2/x] : G'''_2[G''_2/X]$. But the result follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t'_1 : G'''_1[G''_1/X] \leq \Xi_2 \triangleright t'_2 : G'''_2[G''_2/X]$.

Case $(\varepsilon_1 s'_1 :: G''_1 \leq \varepsilon_1 s'_1 :: G''_1)$.

$$\frac{\varepsilon_1 \leq \varepsilon_2 \quad \Omega \vdash \Xi_1 \triangleright s'_1 : G'''_1 \leq \Xi_2 \triangleright s'_2 : G'''_2 \quad G''_1 \sqsubseteq G''_2}{\Omega \vdash \Xi_1 \triangleright \varepsilon_1 s'_1 :: G''_1 : G''_1 \leq \Xi_2 \triangleright \varepsilon_2 s'_2 :: G''_2 : G''_2}$$

Note that we are required to prove that $\Omega \vdash \Xi_1 \triangleright \varepsilon_1 s'_1 :: G''_1 : G''_1 \leq \Xi_2 \triangleright \varepsilon_2 s'_2 :: G''_2 : G''_2$.

$$(\varepsilon_1 s'_1 :: G''_1)[v_1/x] = (\varepsilon_1 s'_1[v_1/x] :: G''_1) \leq (\varepsilon_2 s'_2[v_2/x] :: G''_2) = (\varepsilon_2 s'_2 :: G''_2)[v_2/x]$$

or what is the same $\Omega \vdash \Xi_1 \triangleright s'_1[v_1/x] : G'''_1 \leq \Xi_2 \triangleright s'_2[v_2/x] : G'''_2$. But the result follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright s'_1 : G'''_1 \leq \Xi_2 \triangleright s'_2 : G'''_2$.

Case $(\varepsilon_{G'_1} t'_1 :: G_1 \leq \varepsilon_{G'_2} t'_2 :: G'_2)$. We know that

$$\frac{\varepsilon_{G_1} \not\leq \varepsilon_{G_2} \quad \Omega \vdash \Xi_1 \triangleright t'_1 : G'''_1 \leq \Xi_2 \triangleright t'_2 : G'''_2 \quad G'_1 \sqsubseteq G'_2 \quad G''_1 \rightarrow G'_1 \quad G''_2 \rightarrow G'_2}{\Omega \vdash \Xi_1 \triangleright \varepsilon_{G'_1} t'_1 :: G'_1 : G'_1 \leq \Xi_2 \triangleright \varepsilon_{G'_2} t'_2 :: G'_2 : G'_2}$$

Note that we are required to prove that

$$(\varepsilon_{G'_1} t'_1 :: G'_1)[v_1/x] = (\varepsilon_{G'_1} t'_1[v_1/x] :: G'_1) \leq$$

$$(\varepsilon_{G'_2} t'_2[v_2/x] :: G'_2) = (\varepsilon_{G'_2} t'_2 :: G'_2)[v_2/x]$$

or what is the same $\Omega \vdash \Xi_1 \triangleright t'_1[v_1/x] : G'''_1 \leq \Xi_2 \triangleright t'_2[v_2/x] : G'''_2$. But the result follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t'_1 : G'''_1 \leq \Xi_2 \triangleright t'_2 : G'''_2$.

□

Proposition A.61 *If $G_1^* \leq G_2^*$ then $G_1^* \sqsubseteq G_2^*$.*

PROOF. Examining \leq rules.

Case ($B \leq B$). The results follows immediately by the rule $G \sqsubseteq G$.

Case ($X \leq X$). The results follows immediately by the rule $G \sqsubseteq G$.

Case ($\alpha \leq \alpha$). The results follows immediately by the rule $G \sqsubseteq G$.

Case ($B \leq ?$). The results follows immediately by the rule $G \sqsubseteq ?$.

Case ($G_1 \rightarrow G_2 \leq ?$). The results follows immediately by the rule $G \sqsubseteq ?$.

Case ($G_1 \times G_2 \leq ?$). The results follows immediately by the rule $G \sqsubseteq ?$.

Case ($? \leq ?$). The results follows immediately by the rule $G \sqsubseteq ?$.

Case ($\forall X.G_1 \leq \forall X.G_2$). We know that

$$\frac{G_1 \leq G_2}{\forall X.G_1 \leq \forall X.G_2}$$

By the induction hypothesis on $G_1 \leq G_2$, we know that $G_1 \sqsubseteq G_2$. We are required to prove that $\forall X.G_1 \sqsubseteq \forall X.G_2$, which follows immediately by the rule

$$\frac{G_1 \sqsubseteq G_2}{\forall X.G_1 \sqsubseteq \forall X.G_2}$$

Case ($G_1 \rightarrow G_2 \leq G_3 \rightarrow G_4$). We know that

$$\frac{G_1 \leq G_3 \quad G_2 \leq G_4}{G_1 \rightarrow G_2 \leq G_3 \rightarrow G_4}$$

By the induction hypothesis on $G_1 \leq G_3$ and $G_2 \leq G_4$, we know that $G_1 \sqsubseteq G_3$ and $G_2 \sqsubseteq G_4$. We are required to prove that $G_1 \rightarrow G_2 \sqsubseteq G_3 \rightarrow G_4$, which follows immediately by the rule

$$\frac{G_1 \sqsubseteq G_3 \quad G_2 \sqsubseteq G_4}{G_1 \rightarrow G_2 \sqsubseteq G_3 \rightarrow G_4}$$

Case ($G_1 \times G_2 \leq G_3 \times G_4$). We know that

$$\frac{G_1 \leq G_3 \quad G_2 \leq G_4}{G_1 \times G_2 \leq G_3 \times G_4}$$

By the induction hypothesis on $G_1 \leq G_3$ and $G_2 \leq G_4$, we know that $G_1 \sqsubseteq G_3$ and $G_2 \sqsubseteq G_4$. We are required to prove that $G_1 \times G_2 \sqsubseteq G_3 \times G_4$, which follows immediately by the rule

$$\frac{G_1 \sqsubseteq G_3 \quad G_2 \sqsubseteq G_4}{G_1 \times G_2 \sqsubseteq G_3 \times G_4}$$

□

Proposition A.62 *If $v_1 \leq t_2$ then $t_2 = v_2$.*

PROOF. Exploring \leq rules.

□

Proposition A.63 *If $\varepsilon_1 \leq \varepsilon_2$ then*

- $dom(\varepsilon_1) \leq dom(\varepsilon_2)$
- $cod(\varepsilon_1) \leq cod(\varepsilon_2)$

- $p_i(\varepsilon_1) \leq p_i(\varepsilon_2)$
- $\text{schm}_u(\varepsilon_1) \leq \text{schm}_u(\varepsilon_2)$

PROOF. By inspecting the evidence shape and the definition of $\varepsilon_1 \leq \varepsilon_2$. \square

Proposition A.64 *If $\varepsilon \Vdash \Xi; \Delta \vdash G'' \sim G'$ and $G' \rightarrow G$, then $\varepsilon \circ \varepsilon_G = \varepsilon$, where $\varepsilon_G = \mathcal{I}(\Xi, ()G, G)$.*

PROOF. By Lemma A.115 and definition of $G' \rightarrow G$ and $\varepsilon \circ \varepsilon_G = \varepsilon$. \square

Proposition A.65 *If $\Xi_1 \vdash t_1 \leq \Xi_2 \vdash t_2$ and $\Xi_1 \triangleright t_1 \longrightarrow \Xi'_1 \triangleright t'_1$, then $\Xi_2 \triangleright t_2 \longrightarrow \Xi'_2 \triangleright t'_2$ and $\Xi'_1 \vdash t'_1 \leq \Xi'_2 \vdash t'_2$.*

PROOF. If $\Xi_1 \vdash t_1 \leq \Xi_2 \vdash t_2$, we know that $\vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$, $\Xi_1 \leq \Xi_2$, $\Xi_1 \vdash t_1 : G_1$ and $\Xi_2 \vdash t_2 : G_2$, for some G_1 and G_2 . We follow by induction on $\vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$. We avoid the notation $\vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$, and use $t_1 \leq t_2$ instead, for simplicity, when the typing environments are not relevant.

Case ($\leq b_\varepsilon$). This case does not applies because b is not a term t , therefore it can not reduce.

Case ($\leq x_\varepsilon$). This case does not applies because x can not reduce.

Case ($\leq \lambda_\varepsilon$). This case does not applies because $\lambda x : G_1^*.t_1^*$ is not a term t , therefore it can not reduce.

Case ($\leq \Lambda_\varepsilon$). This case does not applies because $\Lambda X.t_1^*$ is not a term t , therefore it can not reduce.

Case ($\leq \text{app}_\varepsilon$). $t_{11}^* t_{12}^* \leq t_{21}^* t_{22}^*$. We know that

$$\frac{t_{11}^* \leq t_{21}^* \quad t_{12}^* \leq t_{22}^*}{t_{11}^* t_{12}^* \leq t_{21}^* t_{22}^*}$$

Also, since $\Xi_1 \triangleright t_1 \longrightarrow \Xi'_1 \triangleright t'_1$, we know that $t_{11}^* = \varepsilon_{11} \lambda x : G_1^*.t_{11} :: G_{12} \rightarrow G_{11}$ and $t_{12}^* = v_{12} = \varepsilon_{12} u_{12} : G_{12}$. By Proposition A.62, we know that $t_{21}^* = \varepsilon_{21} \lambda x : G_2^*.t_{21} :: G_{22} \rightarrow G_{21}$ and $t_{22}^* = v_{22} = \varepsilon_{22} u_{22} : G_{22}$. By the reduction rules, we know that

$$\Xi_1 \triangleright (\varepsilon_{11} \lambda x : G_1^*.t_{11} :: G_{12} \rightarrow G_{11}) (\varepsilon_{12} u_{12} : G_{12}) \longrightarrow \Xi_1 \triangleright \text{cod}(\varepsilon_{11})(t_{11} [((\varepsilon_{12} \circ \text{dom}(\varepsilon_{11}))u_{11} :: G_1^*)/x]) :: G_{11}$$

By Proposition A.63, we know that $\text{dom}(\varepsilon_{11}) \leq \text{dom}(\varepsilon_{21})$ and $\text{cod}(\varepsilon_{11}) \leq \text{cod}(\varepsilon_{21})$. Therefore, by Proposition 3.20 and $\varepsilon_{12} \leq \varepsilon_{22}$, we know that $(\varepsilon_{12} \circ \text{dom}(\varepsilon_{11})) \leq (\varepsilon_{22} \circ \text{dom}(\varepsilon_{21}))$.

Therefore, we know that

$$\Xi_2 \triangleright (\varepsilon_{21} \lambda x : G_2^*.t_{21} :: G_{22} \rightarrow G_{21}) (\varepsilon_{22} u_{22} : G_{22}) \longrightarrow \Xi_2 \triangleright \text{cod}(\varepsilon_{21})(t_{21} [((\varepsilon_{22} \circ \text{dom}(\varepsilon_{21}))u_{21} :: G_2^*)/x]) :: G_{21}$$

Thus, by the \leq rules, $u_{11} \leq u_{21}$ and $G_1^* \sqsubseteq G_2^*$, we know that

$$((\varepsilon_{12} \circ \text{dom}(\varepsilon_{11}))u_{11} :: G_1^*) \leq ((\varepsilon_{22} \circ \text{dom}(\varepsilon_{21}))u_{21} :: G_2^*)$$

By Proposition A.60, we know that

$$(t_{11}[\!(\varepsilon_{12} \circledast \text{dom}(\varepsilon_{11}))u_{11} :: G_1^*/x]) \leq (t_{21}[\!(\varepsilon_{22} \circledast \text{dom}(\varepsilon_{21}))u_{21} :: G_2^*/x])$$

Finally, since $\text{cod}(\varepsilon_{11}) \leq \text{cod}(\varepsilon_{21})$ and $G_{11} \sqsubseteq G_{21}$ and the \leq rules the result holds.

$$\Xi_1 \vdash \text{cod}(\varepsilon_{11})(t_{11}[\!(\varepsilon_{12} \circledast \text{dom}(\varepsilon_{11}))u_{11} :: G_1^*/x]) :: G_{11} \leq \Xi_2 \vdash \text{cod}(\varepsilon_{21})(t_{21}[\!(\varepsilon_{22} \circledast \text{dom}(\varepsilon_{21}))u_{21} :: G_2^*/x]) :: G_{21}$$

Case ($(\leq_{\text{app}G_\varepsilon}) t_1^* [G_1^*] \leq t_2^* [G_2^*]$). We know that

$$\frac{t_1^* \leq t_2^* \quad G_1^* \leq G_2^*}{t_1^* [G_1^*] \leq t_2^* [G_2^*]}$$

Also, since $\Xi_1 \triangleright t_1 \longrightarrow \Xi'_1 \triangleright t'_1$, we know that $t_1^* = \varepsilon_{11} \Lambda X. t_{11} :: \forall X. G_{11}$. By Proposition A.62, we know that $t_2^* = \varepsilon_{22} \Lambda X. t_{22} :: \forall X. G_{22}$. By the reduction rules, we know that

$$\Xi_1 \triangleright (\varepsilon_{11} \Lambda X. t_{11} :: \forall X. G_{11}) [G_1^*] \longrightarrow \Xi'_1 \triangleright \varepsilon_{11 \text{out}} (\varepsilon_{11} [\hat{\alpha}_1] t_{11} [\hat{\alpha}_1 / X] :: G_{11} [\alpha / X]) :: G_{11} [G_1^* / X]$$

where $\Xi'_1 = \Xi_1$, $\alpha := G_1^*$ and $\hat{\alpha}_1 = \text{lift}_{\Xi'_1}(\alpha)$.

By Proposition A.54, we know that $\varepsilon_{11 \text{out}} \leq \varepsilon_{22 \text{out}}$ and $\varepsilon_{11} [\hat{\alpha}_1] \leq \varepsilon_{22} [\hat{\alpha}_2]$.

Therefore, we know that

$$\Xi_2 \triangleright (\varepsilon_{22} \Lambda X. t_{22} :: \forall X. G_{22}) [G_2^*] \longrightarrow \Xi'_2 \triangleright \varepsilon_{22 \text{out}} (\varepsilon_{22} [\hat{\alpha}_2] t_{22} [\hat{\alpha}_2 / X] :: G_{22} [\alpha / X]) :: G_{22} [G_2^* / X]$$

where $\Xi'_2 = \Xi_2$, $\alpha := G_2^*$ and $\hat{\alpha}_2 = \text{lift}_{\Xi'_2}(\alpha)$.

By Proposition A.59 we know that $t_{11} [\hat{\alpha}_1 / X] \leq t_{22} [\hat{\alpha}_2 / X]$. By Proposition A.55 and Proposition A.56, we know that $G_{11} [\alpha / X] \leq G_{22} [\alpha / X]$ and $G_{11} [G_1^* / X] \leq G_{22} [G_2^* / X]$, respectively.

Finally, by the \leq rules the result holds.

$$\Xi'_1 \triangleright \varepsilon_{1 \text{out}} (\varepsilon_1 [\hat{\alpha}_1] t_1 [\hat{\alpha}_1 / X] :: G_1 [\alpha / X]) :: G_1 [G_1^* / X] \leq \Xi'_2 \triangleright \varepsilon_{2 \text{out}} (\varepsilon_2 [\hat{\alpha}_2] t_2 [\hat{\alpha}_2 / X] :: G_2 [\alpha / X]) :: G_2 [G_2^* / X]$$

Case ($(\leq_{\text{asc}_\varepsilon}) \varepsilon_1 s_1 :: G_1^* \leq \varepsilon_2 s_2 :: G_2^*$). We know that

$$\frac{\varepsilon_1 \leq \varepsilon_2 \quad s_1 \leq s_2 \quad G_1^* \sqsubseteq G_2^*}{\varepsilon_1 s_1 :: G_1^* \leq \varepsilon_2 s_2 :: G_2^*}$$

Also, since $\Xi_1 \triangleright t_1 \longrightarrow \Xi'_1 \triangleright t'_1$, we know that $s_1 = (\varepsilon_{11} u_{11} :: G_{11})$. By Proposition A.62, we know that $s_2 = (\varepsilon_2 u_2 :: G_2)$. By the reduction rules, we know that

$$\Xi_1 \triangleright \varepsilon_1 (\varepsilon_{11} u_{11} :: G_{11}) :: G_1^* \longrightarrow \Xi_1 \triangleright (\varepsilon_{11} \circledast \varepsilon_1) u_{11} :: G_1^*$$

By the \leq rules, we know that $\varepsilon_{11} \leq \varepsilon_{22}$ and $\varepsilon_1 \leq \varepsilon_2$. Therefore, by Proposition 3.20, we know that $(\varepsilon_{11} \circledast \varepsilon_1) \leq (\varepsilon_{22} \circledast \varepsilon_2)$.

Therefore, we know that

$$\Xi_2 \triangleright \varepsilon_2(\varepsilon_{22}u_{22} :: G_{22}) :: G_2^* \longrightarrow \Xi_2 \triangleright (\varepsilon_{22} \circledast \varepsilon_2)u_{22} :: G_2^*$$

Thus, by the \leq rules, $u_{11} \leq u_{22}$ and $G_1^* \sqsubseteq G_2^*$, the result holds.

$$\Xi_1 \vdash (\varepsilon_{11} \circledast \varepsilon_1)u_{11} :: G_1^* \leq \Xi_2 \vdash (\varepsilon_{22} \circledast \varepsilon_2)u_{22} :: G_2^*$$

Case ($(\leq \text{Masc}_\varepsilon)$ $\varepsilon_{G_1^*}t_1^* :: G_1^* \leq \varepsilon_{G_2^*}t_2^* :: G_2^*$). Since $\Xi_1 \triangleright t_1 \longrightarrow \Xi_1' \triangleright t_1'$, we know that $t_1 = \varepsilon_{G_1^*}(\varepsilon_{11}u_{11} :: G_1^{**}) :: G_1^*$, where $t_1^* = (\varepsilon_{11}u_{11} :: G_1^{**})$. By Proposition A.62, we know that $t_2 = \varepsilon_{G_2^*}(\varepsilon_{22}u_{22} :: G_2^{**}) :: G_2^*$, where $t_2^* = (\varepsilon_{22}u_{22} :: G_2^{**})$. We know by rules $\leq \text{asc}_\varepsilon$ and $\leq \text{Masc}_\varepsilon$ that

$$\frac{\begin{array}{c} \varepsilon_{G_1^*} \sqsubseteq \varepsilon_{G_2^*} \quad \varepsilon_{G_1^*} = \mathcal{G}(\Xi_1, ()G_1^*, G_1^*) \quad \varepsilon_{G_2^*} = \mathcal{G}(\Xi_2, ()G_2^*, G_2^*) \quad G_1^{**} \rightarrow G_1^* \quad G_2^{**} \rightarrow G_2^* \\ \varepsilon_{11} \leq \varepsilon_{22} \quad \Omega \vdash \Xi_1 \triangleright u_{11} :: G_1^{**} \leq \Xi_2 \triangleright u_{22} :: G_2^{**} \quad G_1^* \sqsubseteq G_2^* \quad G_1^{**} \sqsubseteq G_2^{**} \end{array}}{\Omega \vdash \Xi_1 \triangleright \varepsilon_{G_1^*}(\varepsilon_{11}u_{11} :: G_1^{**}) :: G_1^* : G_1^* \leq \Xi_2 \triangleright \varepsilon_{G_2^*}(\varepsilon_{22}u_{22} :: G_2^{**}) :: G_2^* : G_2^*}$$

By the reduction rules, we know that

$$\Xi_1 \triangleright \varepsilon_{G_1^*}(\varepsilon_{11}u_{11} :: G_1^{**}) :: G_1^* \longrightarrow \Xi_1 \triangleright (\varepsilon_{11} \circledast \varepsilon_{G_1^*})u_{11} :: G_1^*$$

We know by the definition of \vdash $\Xi_1 \triangleright \varepsilon_{G_1^*}(\varepsilon_{11}u_{11} :: G_1^{**}) :: G_1^* : G_1^* \leq \Xi_2 \triangleright \varepsilon_{G_2^*}(\varepsilon_{22}u_{22} :: G_2^{**}) :: G_2^* : G_2^*$ that $\Xi_1 \vdash (\varepsilon_{11}u_{11} :: G_1^{**}) : G_1^{**}$ and $\Xi_2 \vdash (\varepsilon_{22}u_{22} :: G_2^{**}) : G_2^{**}$, and therefore, $\varepsilon_{11} \Vdash \Xi_1 \vdash G_1^{**} \sim G_1^*$ and $\varepsilon_{22} \Vdash \Xi_2 \vdash G_2^{**} \sim G_2^*$. We know that $\varepsilon_{11} \leq \varepsilon_{22}$ and $\varepsilon_{G_1^*} \sqsubseteq \varepsilon_{G_2^*}$. Therefore, by Lemma A.64 and $G_1^{**} \rightarrow G_1^*$ and $G_2^{**} \rightarrow G_2^*$, we know that $(\varepsilon_{11} \circledast \varepsilon_{G_1^*}) = \varepsilon_{11}$ and $(\varepsilon_{22} \circledast \varepsilon_{G_2^*}) = \varepsilon_{22}$.

Therefore, we know that

$$\Xi_2 \triangleright \varepsilon_{G_2^*}(\varepsilon_{22}u_{22} :: G_2^{**}) :: G_2^* \longrightarrow \Xi_2 \triangleright (\varepsilon_{22} \circledast \varepsilon_{G_2^*})u_{22} :: G_2^*$$

Then, by the \leq rules, $u_{11} \leq u_{22}$ and $G_1^* \sqsubseteq G_2^*$, the result holds.

$$\Xi_1 \vdash (\varepsilon_{11} \circledast \varepsilon_1)u_{11} :: G_1^* \leq \Xi_2 \vdash (\varepsilon_{22} \circledast \varepsilon_2)u_{22} :: G_2^*$$

□

Proposition A.66 *If $\Xi_1 \vdash t_1 \leq \Xi_2 \vdash t_2$ and $\Xi_1 \triangleright t_1 \mapsto \Xi_1' \triangleright t_1'$, then $\Xi_2 \triangleright t_2 \mapsto \Xi_2' \triangleright t_2'$ and $\Xi_1' \vdash t_1' \leq \Xi_2' \vdash t_2'$.*

PROOF. If $\Xi_1 \vdash t_1 \leq \Xi_2 \vdash t_2$, we know that $\vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$, $\Xi_1 \leq \Xi_2$, $\Xi_1 \vdash t_1 : G_1$ and $\Xi_2 \vdash t_2 : G_2$. We avoid the notation $\vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$, and use $t_1 \leq t_2$ instead, for simplicity, when the typing environments are not relevant.

By induction on reduction $\Xi_1 \triangleright t_1 \mapsto \Xi_1' \triangleright t_1'$.

Case ($\Xi_1 \triangleright t_1 \longrightarrow \Xi_1' \triangleright t_1'$). By Proposition A.65, we know that $\Xi_2 \triangleright t_2 \longrightarrow \Xi_2' \triangleright t_2'$, $\Xi_1' \vdash t_1' \leq \Xi_2' \vdash t_2'$; and the result holds immediately.

Case $(\Xi_1 \triangleright \varepsilon_{11} t_{11} :: G_{11} \mapsto \Xi'_1 \triangleright \varepsilon_{11} t'_{11} :: G_{11})$. By inspection of \leq , $t_2 = \varepsilon_{22} t_{22} :: G_{22}$, where $\varepsilon_{11} \leq \varepsilon_{22}$ or $\varepsilon_{11} \sqsubseteq \varepsilon_{22}$, $t_{11} \leq t_{22}$ and $G_{11} \sqsubseteq G_{22}$. By induction hypothesis on $\Xi_1 \triangleright t_{11} \mapsto \Xi'_1 \triangleright t'_{11}$, then $\Xi_2 \triangleright t_{22} \mapsto \Xi'_2 \triangleright t'_{22}$, where $\Xi'_1 \vdash t'_{11} \leq \Xi'_2 \vdash t'_{22}$. Then, by \leq , we know that $\Xi'_1 \vdash \varepsilon_{11} t'_{11} :: G_{11} \leq \Xi'_2 \vdash \varepsilon_{22} t'_{22} :: G_{22}$ and the result holds.

Case $(\Xi_1 \triangleright t_{11} t_{12} \mapsto \Xi'_1 \triangleright t'_{11} t_{12})$. By inspection of \leq , $t_2 = t_{21} t_{22}$, where $t_{11} \leq t_{21}$ and $t_{12} \leq t_{22}$. By induction hypothesis on $\Xi_1 \triangleright t_{11} \mapsto \Xi'_1 \triangleright t'_{11}$, we know that $\Xi_2 \triangleright t_{21} \mapsto \Xi'_2 \triangleright t'_{21}$, where $\Xi'_1 \vdash t'_{11} \leq \Xi'_2 \vdash t'_{21}$. Then, by \leq , we know that $\Xi'_1 \vdash t'_{11} t_{12} \leq \Xi'_2 \vdash t'_{21} t_{22}$ and the result holds.

Case $(\Xi_1 \triangleright v_{11} t_{12} \mapsto \Xi'_1 \triangleright v_{11} t'_{12})$. By inspection of \leq and Proposition A.62, $t_2 = v_{21} t_{22}$, where $v_{11} \leq v_{21}$ and $t_{12} \leq t_{22}$. By induction hypothesis on $\Xi_1 \triangleright t_{12} \mapsto \Xi'_1 \triangleright t'_{12}$, then $\Xi_2 \triangleright t_{22} \mapsto \Xi'_2 \triangleright t'_{22}$, where $\Xi'_1 \vdash t'_{12} \leq \Xi'_2 \vdash t'_{22}$. Then, by \leq , we know that $\Xi'_1 \vdash v_{11} t'_{12} \leq \Xi'_2 \vdash v_{21} t'_{22}$ and the result holds.

Case $(\Xi_1 \triangleright t_{11} [G_{11}] \mapsto \Xi'_1 \triangleright t'_{11} [G_{11}])$. By inspection of \leq , $t_2 = t_{22} [G_{22}]$, where $t_{11} \leq t_{22}$ and $G_{11} \leq G_{22}$. By induction hypothesis on $\Xi_1 \triangleright t_{11} \mapsto \Xi'_1 \triangleright t'_{11}$, we know that $\Xi_2 \triangleright t_{22} \mapsto \Xi'_2 \triangleright t'_{22}$, where $\Xi'_1 \vdash t'_{11} \leq \Xi'_2 \vdash t'_{22}$. Then, by \leq , we know that $\Xi'_1 \vdash t'_{11} [G_{11}] \leq \Xi'_2 \vdash t'_{22} [G_{22}]$ and the result holds. □

Proposition A.67 (Small-step DGG $^{\leq}$ for GSF ε) *Suppose $\Xi_1 \triangleright t_1 \leq \Xi_2 \triangleright t_2$.*

- a. *If $\Xi_1 \triangleright t_1 \mapsto \Xi'_1 \triangleright t'_1$, then $\Xi_2 \triangleright t_2 \mapsto \Xi'_2 \triangleright t'_2$, for some Ξ'_2 and t'_2 such that $\Xi'_1 \triangleright t'_1 \leq \Xi'_2 \triangleright t'_2$.*
- b. *If $t_1 = v_1$, then $t_2 = v_2$.*

PROOF. Direct by Lemma A.66 and A.62. □

Proposition A.68 *Let suppose $\Xi_1 \vdash t_1 \leq \Xi_2 \vdash t_2$.*

- $\Xi_1 \triangleright t_1 \mapsto^* \Xi'_1 \triangleright v_1$ implies $\Xi_2 \triangleright t_2 \mapsto^* \Xi'_2 \triangleright v_2$, $\Xi'_1 \vdash v_1 \leq \Xi'_2 \vdash v_2$.
- t_1 diverges implies t_2 diverges.
- $\Xi_2 \triangleright t_2 \mapsto^* \Xi'_2 \triangleright v_2$ implies $\Xi_1 \triangleright t_1 \mapsto^* \Xi'_1 \triangleright v_1$ and $\Xi'_1 \vdash v_1 \leq \Xi'_2 \vdash v_2$, or $\Xi_1 \triangleright t_1 \mapsto^*$ **error**.
- t_2 diverges implies t_1 diverges, or $\Xi_1 \triangleright t_1 \mapsto^*$ **error**.

PROOF. The proof is by case analysis on the reduction of t_1 or t_2 .

- Suppose that $\Xi_1 \triangleright t_1 \mapsto^* \Xi'_1 \triangleright v_1$. Then $\Xi_2 \triangleright t_2 \mapsto^* \Xi'_2 \triangleright v_2$, $\Xi'_1 \vdash v_1 \leq \Xi'_2 \vdash v_2$ by Proposition A.66 and Proposition A.62.
- Suppose that t_1 diverges. Then t_2 diverges by Proposition A.66.
- Suppose that $\Xi_2 \triangleright t_2 \mapsto^* \Xi'_2 \triangleright v_2$. Then, the only possibilities given the two previous results are $\Xi_1 \triangleright t_1 \mapsto^* \Xi'_1 \triangleright v_1$ and $\Xi'_1 \vdash v_1 \leq \Xi'_2 \vdash v_2$, or $\Xi_1 \triangleright t_1 \mapsto^*$ **error**, and the result holds.
- Suppose that t_2 diverges. Then, the only possibilities given the two previous results are t_1 diverges, or $\Xi_1 \triangleright t_1 \mapsto^*$ **error**, and the result holds.

□

Theorem A.69 (DGG[≤]) *Suppose $t_1 \leq t_2$, $\vdash t_1 : G_1$, and $\vdash t_2 : G_2$.*

- a. *If $t_1 \Downarrow \Xi_1 \triangleright v_1$, then $t_2 \Downarrow \Xi_2 \triangleright v_2$, $\cdot \vdash \Xi_1 \triangleright v_1 : G_1 \leq \Xi_2 \triangleright v_2 : G_2$ and $\Xi_1 \leq \Xi_2$, for some v_2 and Ξ_2 .
If $t_1 \Uparrow$ then $t_2 \Uparrow$.*
- b. *If $t_2 \Downarrow \Xi_2 \triangleright v_2$, then $t_1 \Downarrow \Xi_1 \triangleright v_1$, $\cdot \vdash \Xi_1 \triangleright v_1 : G_1 \leq \Xi_2 \triangleright v_2 : G_2$ and $\Xi_1 \leq \Xi_2$, for some v_1 and Ξ_1 , or $t_1 \Downarrow$ **error**.
If $t_2 \Uparrow$, then $t_1 \Uparrow$ or $t_1 \Downarrow$ **error**.*

PROOF. Direct by Lemma A.80 and A.68. □

Lemma A.70 *Let $\vdash t : G$, $G \sqsubseteq G'$, and $t' = t :: G' :: G$, then*

- $t \Downarrow \Xi \triangleright v \iff t' \Downarrow \Xi \triangleright v$.
- $t \Downarrow$ **error** $\iff t' \Downarrow$ **error**.

PROOF. Direct consequence of the weak dynamic gradual guarantee (Theorem 3.22). □

Lemma A.71 *Let $\vdash t_1 : G_1$ and $\vdash t_2 : G_2$ such that $\vdash t_1 t_2 : G$ and $t_1 t_2 \Downarrow \Xi \triangleright v$. Let $G_1 \sqsubseteq G'_1$, $G_2 \sqsubseteq G'_2$, and $G \sqsubseteq G'$, such that $\vdash (t_1 :: G'_1) (t_2 :: G'_2) : G'$. Then $(t_1 :: G'_1) (t_2 :: G'_2) \Downarrow \Xi \triangleright v'$ such that $\vdash \Xi \triangleright v : G \leq \Xi \triangleright v' : G'$, for some v' .*

PROOF. From $\vdash (t_1 :: G'_1) (t_2 :: G'_2) : G$, we know that $\vdash G_1 \sim G'_1$ and $\vdash G_2 \sim G'_2$, where $\vdash t_1 : G_1$ and $\vdash t_2 : G_2$. As $G_1 \sqsubseteq G'_1$, and $G_2 \sqsubseteq G'_2$, then $G_1 \text{pullback} G_1 \sqsubseteq G_1 \text{pullback} G'_1$ and $G_2 \text{pullback} G_2 \sqsubseteq G_2 \text{pullback} G'_2$. Notice that if $t_1 t_2 \Downarrow v$, then $(t_1 :: G_1) (t_2 :: G_2) \Downarrow v$ (trivial ascriptions). Therefore, by (\leq ascv) or (\leq asct), $\vdash (t_1 :: G_1) (t_2 :: G_2) : T \leq (t_1 :: G'_1) (t_2 :: G'_2) : G$, then the result holds by DGG[≤] (Th.3.22). □

Lemma A.72 *Let $\vdash t : G$ such that $t \Downarrow \Xi \triangleright v$, and $G \sqsubseteq G'$. Then $t :: G' \Downarrow \Xi \triangleright v'$ such that $\vdash \Xi \triangleright v : G \leq \Xi \triangleright v' : G'$, for some v' .*

Direct by Th.3.22. Similar to Lemma 3.25.

Lemma A.73 *Let $\vdash t : G_1$ such that $\vdash t [G_2] : G$ and $t [G_2] \Downarrow \Xi \triangleright v$. Let $G_1 \sqsubseteq G'_1$, $G_2 \leq G'_2$, and $G \sqsubseteq G'$, such that $\vdash (t :: G'_1) [G'_2] : G'$. Then $(t :: G'_1) [G'_2] \Downarrow \Xi' \triangleright v'$ such that $\vdash \Xi \triangleright v : G \leq \Xi' \triangleright v' : G'$ and $\Xi \leq \Xi'$, for some v' and Ξ' .*

PROOF. Direct by Th.3.22. Similar to Lemma 3.25. □

Proposition A.74 *Suppose t_1 and t_2 GSF terms such that $\cdot \vdash t_1 : G_1 \leq t_2 : G_2$, and their elaborations $\cdot \vdash t_1 \rightsquigarrow t_{\varepsilon_1} : G_1$ and $\cdot \vdash t_2 \rightsquigarrow t_{\varepsilon_2} : G_2$. Then $\cdot \vdash \triangleright t_{\varepsilon_1} : G_1 \leq \triangleright t_{\varepsilon_2} : G_2$.*

PROOF. Direct by Prop. A.80. □

A.5.4 Syntactic Strict Precision for GSF

Now, we present the proof of the weak dynamic gradual guarantee for GSF previously presented and the auxiliary Propositions and Definitions.

Proposition A.75 $\mathcal{I}(\Xi, ()G_1 \text{pullback} G_2, G_1 \text{pullback} G_2) = \mathcal{I}(\Xi, ()G_1, G_2)$

PROOF. By the definition of *pullback* and $\mathcal{I}(\Xi, ()G_1, G_2)$. □

Proposition A.76 $\Omega \vdash \Xi_1 \triangleright s_1 : G_1 \leq \Xi_2 \triangleright s_2 : G_2$ then $G_1 \sqsubseteq G_2$.

PROOF. By the definition of *pullback* and $\mathcal{I}(\Xi, ()G_1, G_2)$. □

Proposition A.77 If $G_1 \text{pullback} G_2 \leq G'_1 \text{pullback} G'_2$, then

$$\mathcal{I}(\Xi, ()G_1, G_2) = \mathcal{I}(\Xi, ()G_1 \text{pullback} G_2, G_1 \text{pullback} G_2) \leq \mathcal{I}(\Xi, ()G'_1 \text{pullback} G'_2, G'_1 \text{pullback} G'_2) = \mathcal{I}(\Xi, ()G'_1, G'_2)$$

PROOF. By Proposition A.75 and the definition of \leq in evidence. □

Proposition A.78 If $G_1 \leq G_2$, then

$$\mathcal{I}(\Xi, ()G_1, G_1) \sqsubseteq \mathcal{I}(\Xi, ()G_2, G_2)$$

PROOF. By the definition of $\mathcal{I}(\Xi, \cdot)$ and the \sqsubseteq in evidence. □

Definition A.79 $\Omega \equiv \Gamma_1 \sqsubseteq \Gamma_2 \iff (\Omega = \Omega', x : G_1 \sqsubseteq G_2, \Gamma_1 = \Gamma'_1, x : G_1, \Gamma_2 = \Gamma'_2, x : G_2, G_1 \sqsubseteq G_2 \text{ and } \Omega' \equiv \Gamma'_1 \sqsubseteq \Gamma'_2) \vee (\Omega = \Gamma_1 = \Gamma_2 = \cdot)$.

Proposition A.80 If $\Omega \vdash \Xi_1 \triangleright t_1^* : G_1^* \leq \Xi_2 \triangleright t_2^* : G_2^*$, $\Omega \equiv \Gamma_1 \sqsubseteq \Gamma_2$, $\Xi_1 \leq \Xi_2$ and $\Xi_i; \Delta; \Gamma_i \vdash t_i^* \rightsquigarrow t_i^{**} : G_i^*$, then $\Omega \vdash \Xi_1 \triangleright t_1^{**} : G_1^* \leq \Xi_2 \triangleright t_2^{**} : G_2^*$.

PROOF. We follow by induction on $\Omega \vdash \Xi_1 \triangleright t_1^* : G_1^* \leq \Xi_2 \triangleright t_2^* : G_2^*$. We avoid the notation $\Omega \vdash \Xi_1 \triangleright t_1^* : G_1^* \leq \Xi_2 \triangleright t_2^* : G_2^*$, and use $t_1^* \leq t_2^*$ instead, for simplicity, when the typing environments are not relevant. We use metavariable v or u in GSF to range over constants, functions and type abstractions.

Remember that

$$\text{norm}(t, G_1, G_2) = \varepsilon t :: G_2, \text{ where } \varepsilon = \mathcal{I}(\Xi, ()G_1, G_2)$$

By Proposition A.75 we know that

$$\mathcal{I}(\Xi, ()G_1, G_2) = \mathcal{I}(\Xi, ()G_1 \text{pullback} G_2, G_1 \text{pullback} G_2) = \mathcal{I}(\cdot, ()\text{lift}_{\Xi}(G_1), \text{lift}_{\Xi}(G_2))$$

Case $(\Omega \vdash \Xi_1 \triangleright u_1 : G_1^* \leq_v \Xi_2 \triangleright u_2 : G_2^*)$. We know that

$$\begin{aligned}
& (\leq_v) \frac{\Omega \vdash u_1 : G_1^* \leq_v u_2 : G_2^* \quad G_1^* \leq G_2^*}{\Omega \vdash \Xi_1 \triangleright u_1 : G_1^* \leq \Xi_2 \triangleright u_2 : G_2^*} \\
& (\text{Gu}) \frac{\Xi_1; \Delta; \Gamma_1 \vdash u_1 \rightsquigarrow u'_1 : G_1^* \quad \varepsilon_{G_1^*} = \mathcal{I}(\Xi, ()G_1^*, G_1^*)}{\Xi_1; \Delta; \Gamma_1 \vdash u_1 \rightsquigarrow \varepsilon_{G_1^*} u'_1 :: G_1^* : G_1^*} \\
& (\text{Gu}) \frac{\Xi_2; \Delta; \Gamma_2 \vdash u_2 \rightsquigarrow u'_2 : G_2^* \quad \varepsilon_{G_2^*} = \mathcal{I}(\Xi, ()G_2^*, G_2^*)}{\Xi_2; \Delta; \Gamma_2 \vdash u_2 \rightsquigarrow \varepsilon_{G_2^*} u'_2 :: G_2^* : G_2^*}
\end{aligned}$$

We have to prove that $\Omega \vdash \Xi_1 \triangleright \varepsilon_{G_1^*} u'_1 :: G_1^* : G_1^* \leq \Xi_2 \triangleright \varepsilon_{G_2^*} u'_2 :: G_2^* : G_2^*$. By the rule $(\leq_{\text{asc}_\varepsilon})$, we are required to prove that $\varepsilon_{G_1^*} \leq \varepsilon_{G_2^*}$, $\Omega \vdash \Xi_1 \triangleright u'_1 : G_1^* \leq \Xi_2 \triangleright u'_2 : G_2^*$ and $G_1^* \sqsubseteq G_2^*$. Since $G_1^* \leq G_2^*$, $\Xi_1 \leq \Xi_2$ and Proposition A.50, we know that $\varepsilon_{G_1^*} \leq \varepsilon_{G_2^*}$. Also, by Proposition A.61 and $G_1^* \leq G_2^*$ we now that $G_1^* \sqsubseteq G_2^*$. Therefore, we only have required to prove that $\Omega \vdash \Xi_1 \triangleright u'_1 : G_1^* \leq \Xi_2 \triangleright u'_2 : G_2^*$. We follow by case analysis on $\Omega \vdash u_1 : G_1^* \leq_v u_2 : G_2^*$.

- Case $(\Omega \vdash b : B \leq_v b : B)$. We know that

$$\begin{aligned}
& (\leq_b) \frac{ty(b) = B}{\Omega \vdash b : B \leq_v b : B} \\
& (\text{Gb}) \frac{ty(b) = B}{\Xi_i; \Delta; \Gamma_i \vdash b \rightsquigarrow b : B}
\end{aligned}$$

We have to prove that $\Omega \vdash \Xi_1 \triangleright b : B \leq \Xi_2 \triangleright b : B$. Then, by (\leq_{b_ε}) rule, we know that $\Omega \vdash \Xi_1 \triangleright b : B \leq \Xi_2 \triangleright b : B$ and the result holds.

- Case $(\Omega \vdash (\lambda x : G_1.t_1) : G_1 \rightarrow G_2 \leq_v (\lambda x : G'_1.t_2) : G'_1 \rightarrow G'_2)$. We know that

$$\begin{aligned}
& (\leq_\lambda) \frac{\Omega, x : G_1 \sqsubseteq G'_1 \vdash \Xi_1 \triangleright t_1 : G_2 \leq \Xi_2 \triangleright t_2 : G'_2 \quad G_1 \sqsubseteq G'_1}{\Omega \vdash (\lambda x : G_1.t_1) : G_1 \rightarrow G_2 \leq_v (\lambda x : G'_1.t_2) : G'_1 \rightarrow G'_2} \\
& (\text{G}\lambda) \frac{\Xi_1; \Delta; \Gamma_1, x : G_1 \vdash t_1 \rightsquigarrow t'_1 : G_2}{\Xi_1; \Delta; \Gamma_1 \vdash (\lambda x : G_1.t_1) \rightsquigarrow (\lambda x : G_1.t'_1) : G_1 \rightarrow G_2} \\
& (\text{G}\lambda) \frac{\Xi_2; \Delta; \Gamma_2, x : G'_1 \vdash t_2 \rightsquigarrow t'_2 : G'_2}{\Xi_2; \Delta; \Gamma_2 \vdash (\lambda x : G'_1.t_2) \rightsquigarrow (\lambda x : G'_1.t'_2) : G'_1 \rightarrow G'_2}
\end{aligned}$$

Therefore, we are required to prove that $\Omega \vdash \Xi_1 \triangleright (\lambda x : G_1.t'_1) : G_1 \rightarrow G_2 \leq \Xi_2 \triangleright (\lambda x : G'_1.t'_2) : G'_1 \rightarrow G'_2$, or what is the same by the $(\leq_{\lambda_\varepsilon})$ that $\Omega, x : G_1 \sqsubseteq G'_1 \vdash \Xi_1 \triangleright t'_1 : G_2 \leq \Xi_2 \triangleright t'_2 : G'_2$, but the result follows immediately by the induction hypothesis on $\Omega, x : G_1 \sqsubseteq G'_1 \vdash \Xi_1 \triangleright t_1 : G_2 \leq \Xi_2 \triangleright t_2 : G'_2$, with the translations t'_1 and t'_2 ($\Omega, x : G_1 \sqsubseteq G'_1 \equiv \Gamma_1, x : G_1 \sqsubseteq \Gamma_2, x : G'_1$).

- Case $(\Omega \vdash (\Lambda X.t_1) : \forall X.G_1 \leq_v (\Lambda X.t_2) : \forall X.G_2)$. We know that

$$\begin{aligned}
& (\leq_\Lambda) \frac{\Omega \vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2}{\Omega \vdash (\Lambda X.t_1) : \forall X.G_1 \leq_v (\Lambda X.t_2) : \forall X.G_2} \\
& (\text{G}\lambda) \frac{\Xi_1; \Delta, X; \Gamma_1 \vdash t_1 \rightsquigarrow t'_1 : G_1}{\Xi_1; \Delta; \Gamma_1 \vdash (\Lambda X.t_1) \rightsquigarrow (\Lambda X.t'_1) : \forall X.G_1}
\end{aligned}$$

$$(G\lambda) \frac{\Xi_2; \Delta, X; \Gamma_2 \vdash t_2 \rightsquigarrow t'_2 : G_2}{\Xi_2; \Delta; \Gamma_2 \vdash (\Lambda X.t_2) \rightsquigarrow (\Lambda X.t'_2) : \forall X.G_2}$$

Therefore, we are required to prove that $\Omega \vdash \Xi_1 \triangleright (\Lambda X.t'_1) : \forall X.G_1 \leq \Xi_2 \triangleright (\Lambda X.t'_2) : \forall X.G_2$, or what is the same by the rule ($\leq_{\Lambda_\varepsilon}$) that $\Omega \vdash \Xi_1 \triangleright t'_1 : G_1 \leq \Xi_2 \triangleright t'_2 : G_2$, but the result follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$, with the translations t'_1 and t'_2 .

Case ($\Omega \vdash \Xi_1 \triangleright x : G_1^* \leq \Xi_2 \triangleright x : G_2^*$). We know that

$$(\leq x) \frac{x : G_1^* \sqsubseteq G_2^* \in \Omega}{\Omega \vdash \Xi_1 \triangleright x : G_1^* \leq \Xi_2 \triangleright x : G_2^*}$$

$$(Gx) \frac{x : G_1^* \in \Gamma_1}{\Xi_1; \Delta; \Gamma_1 \vdash x \rightsquigarrow x : G_1^*}$$

$$(Gx) \frac{x : G_2^* \in \Gamma_2}{\Xi_2; \Delta; \Gamma_2 \vdash x \rightsquigarrow x : G_2^*}$$

We have to prove that $\Omega \vdash \Xi_1 \triangleright x : G_1^* \leq \Xi_2 \triangleright x : G_2^*$. Then, by the rule (\leq_{x_ε}), we know that $\Omega \vdash \Xi_1 \triangleright x : G_1^* \leq \Xi_2 \triangleright x : G_2^*$ and the result holds.

Case (\leq_{ascv}). We know that

$$(\leq_{\text{ascv}}) \frac{\Omega \vdash u_1 : G_1^{**} \leq_v u_2 : G_2^{**} \quad G_1^{**} \text{ pullback } G_1^* \leq G_2^{**} \text{ pullback } G_2^* \quad G_1^* \sqsubseteq G_2^*}{\Omega \vdash \Xi_1 \triangleright u_1 :: G_1^* : G_1^* \leq \Xi_2 \triangleright u_2 :: G_2^* : G_2^*}$$

$$(G\text{ascu}) \frac{\Xi_1; \Delta; \Gamma_1 \vdash u_1 \rightsquigarrow u'_1 : G_1^{**} \quad \varepsilon_1 = \mathcal{I}(\Xi, ()G_1^{**}, G_1^*)}{\Xi_1; \Delta; \Gamma_1 \vdash u_1 :: G_1^* \rightsquigarrow \varepsilon_1 u'_1 :: G_1^* : G_1^*}$$

$$(G\text{ascu}) \frac{\Xi_2; \Delta; \Gamma_2 \vdash u_2 \rightsquigarrow u'_2 : G_2^{**} \quad \varepsilon_2 = \mathcal{I}(\Xi, ()G_2^{**}, G_2^*)}{\Xi_2; \Delta; \Gamma_2 \vdash u_2 :: G_2^* \rightsquigarrow \varepsilon_2 u'_2 :: G_2^* : G_2^*}$$

We have to prove that $\Omega \vdash \Xi_1 \triangleright \varepsilon_1 u'_1 :: G_1^* : G_1^* \leq \Xi_2 \triangleright \varepsilon_2 u'_2 :: G_2^* : G_2^*$, or what is the same by the rule ($\leq_{\text{asc}_\varepsilon}$), we have to prove that $\varepsilon_1 \leq \varepsilon_2$, $\Omega \vdash \Xi_1 \triangleright u'_1 : G_1^{**} \leq \Xi_2 \triangleright u'_2 : G_2^{**}$ and $G_1^* \sqsubseteq G_2^*$. By Proposition A.75, we know that $\varepsilon_1 = \mathcal{I}(\Xi, ()G_1^{**}, G_1^*) = \mathcal{I}(\Xi, ()G_1^{**} \text{ pullback } G_1^*, G_1^{**} \text{ pullback } G_1^*)$ and $\varepsilon_2 = \mathcal{I}(\Xi, ()G_2^{**}, G_2^*) = \mathcal{I}(\Xi, ()G_2^{**} \text{ pullback } G_2^*, G_2^{**} \text{ pullback } G_2^*)$. Since $G_1^{**} \text{ pullback } G_1^* \leq G_2^{**} \text{ pullback } G_2^*$, then $\varepsilon_1 = \mathcal{I}(\Xi, ()G_1^{**}, G_1^*) = \mathcal{I}(\Xi, ()G_1^{**} \text{ pullback } G_1^*, G_1^{**} \text{ pullback } G_1^*) \leq \mathcal{I}(\Xi, ()G_2^{**} \text{ pullback } G_2^*, G_2^{**} \text{ pullback } G_2^*) = \mathcal{I}(\Xi, ()G_2^{**}, G_2^*) = \varepsilon_2$, by Proposition A.77. Thus, we only have to prove that $\Omega \vdash \Xi_1 \triangleright u'_1 : G_1^{**} \leq \Xi_2 \triangleright u'_2 : G_2^{**}$, and we know that $\Omega \vdash u'_1 : G_1^{**} \leq_v u'_2 : G_2^{**}$. We follow by case analysis on $\Omega \vdash u_1 : G_1^{**} \leq_v u_2 : G_2^{**}$.

- *Case* ($\Omega \vdash b : B \leq_v b : B$). We know that

$$(\leq b) \frac{ty(b) = B}{\Omega \vdash b : B \leq_v b : B}$$

$$(Gb) \frac{ty(b) = B}{\Xi_i; \Delta; \Gamma_i \vdash b \rightsquigarrow b : B}$$

We have to prove that $\Omega \vdash \Xi_1 \triangleright b : B \leq \Xi_2 \triangleright b : B$. Then, by $(\leq b_\varepsilon)$ rule, we know that $\Omega \vdash \Xi_1 \triangleright b : B \leq \Xi_2 \triangleright b : B$ and the result holds.

- Case $(\Omega \vdash (\lambda x : G_1.t_1) : G_1 \rightarrow G_2 \leq_v (\lambda x : G'_1.t_2) : G'_1 \rightarrow G'_2)$. We know that

$$(\leq \lambda) \frac{\Omega, x : G_1 \sqsubseteq G'_1 \vdash \Xi_1 \triangleright t_1 : G_2 \leq \Xi_2 \triangleright t_2 : G'_2 \quad G_1 \sqsubseteq G'_1}{\Omega \vdash (\lambda x : G_1.t_1) : G_1 \rightarrow G_2 \leq_v (\lambda x : G'_1.t_2) : G'_1 \rightarrow G'_2}$$

$$(G\lambda) \frac{\Xi_1; \Delta; \Gamma_1, x : G_1 \vdash t_1 \rightsquigarrow t'_1 : G_2}{\Xi_1; \Delta; \Gamma_1 \vdash (\lambda x : G_1.t_1) \rightsquigarrow (\lambda x : G_1.t'_1) : G_1 \rightarrow G_2}$$

$$(G\lambda) \frac{\Xi_2; \Delta; \Gamma_2, x : G'_1 \vdash t_2 \rightsquigarrow t'_2 : G'_2}{\Xi_2; \Delta; \Gamma_2 \vdash (\lambda x : G'_1.t_2) \rightsquigarrow (\lambda x : G'_1.t'_2) : G'_1 \rightarrow G'_2}$$

Therefore, we are required to prove that $\Omega \vdash \Xi_1 \triangleright (\lambda x : G_1.t'_1) : G_1 \rightarrow G_2 \leq \Xi_2 \triangleright (\lambda x : G'_1.t'_2) : G'_1 \rightarrow G'_2$, or what is the same by the $(\leq \lambda_\varepsilon)$ that $\Omega, x : G_1 \sqsubseteq G'_1 \vdash \Xi_1 \triangleright t'_1 : G_2 \leq \Xi_2 \triangleright t'_2 : G'_2$, but the result follows immediately by the induction hypothesis on $\Omega, x : G_1 \sqsubseteq G'_1 \vdash \Xi_1 \triangleright t_1 : G_2 \leq \Xi_2 \triangleright t_2 : G'_2$, with the translations t'_1 and t'_2 ($\Omega, x : G_1 \sqsubseteq G'_1 \equiv \Gamma_1, x : G_1 \sqsubseteq \Gamma_2, x : G'_1$).

- Case $(\Omega \vdash (\Lambda X.t_1) : \forall X.G_1 \leq_v (\Lambda X.t_2) : \forall X.G_2)$. We know that

$$(\leq \Lambda) \frac{\Omega \vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2}{\Omega \vdash (\Lambda X.t_1) : \forall X.G_1 \leq_v (\Lambda X.t_2) : \forall X.G_2}$$

$$(G\lambda) \frac{\Xi_1; \Delta, X; \Gamma_1 \vdash t_1 \rightsquigarrow t'_1 : G_1}{\Xi_1; \Delta; \Gamma_1 \vdash (\Lambda X.t_1) \rightsquigarrow (\Lambda X.t'_1) : \forall X.G_1}$$

$$(G\lambda) \frac{\Xi_2; \Delta, X; \Gamma_2 \vdash t_2 \rightsquigarrow t'_2 : G_2}{\Xi_2; \Delta; \Gamma_2 \vdash (\Lambda X.t_2) \rightsquigarrow (\Lambda X.t'_2) : \forall X.G_2}$$

Therefore, we are required to prove that $\Omega \vdash \Xi_1 \triangleright (\Lambda X.t'_1) : \forall X.G_1 \leq \Xi_2 \triangleright (\Lambda X.t'_2) : \forall X.G_2$, or what is the same by the rule $(\leq \Lambda_\varepsilon)$ that $\Omega \vdash \Xi_1 \triangleright t'_1 : G_1 \leq \Xi_2 \triangleright t'_2 : G_2$, but the result follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$, with the translations t'_1 and t'_2 .

Case $(\Omega \vdash \Xi_1 \triangleright t_1 :: G_1^* : G_1^* \leq \Xi_2 \triangleright t_2 :: G_2^* : G_2^*)$.

$$(\leq \text{asct}) \frac{\Omega \vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2 \quad G_1 \text{pullback} G_1^* \leq G_2 \text{pullback} G_2^* \quad G_1^* \sqsubseteq G_2^*}{\Omega \vdash \Xi_1 \triangleright t_1 :: G_1^* : G_1^* \leq \Xi_2 \triangleright t_2 :: G_2^* : G_2^*}$$

$$(\text{Gasct}) \frac{\Xi_1; \Delta; \Gamma_1 \vdash t_1 \rightsquigarrow t'_1 : G_1 \quad \varepsilon_1 = \mathcal{I}(\Xi, ())G_1, G_1^*}{\Xi_1; \Delta; \Gamma_1 \vdash t_1 :: G_1^* \rightsquigarrow \varepsilon_1 t'_1 :: G_1^* : G_1^*}$$

$$(\text{Gasct}) \frac{\Xi_2; \Delta; \Gamma_2 \vdash t_2 \rightsquigarrow t'_2 : G_2 \quad \varepsilon_2 = \mathcal{I}(\Xi, ())G_2, G_2^*}{\Xi_2; \Delta; \Gamma_2 \vdash t_2 :: G_2^* \rightsquigarrow \varepsilon_2 t'_2 :: G_2^* : G_2^*}$$

We have to prove that $\Omega \vdash \Xi_1 \triangleright \varepsilon_1 t'_1 :: G_1^* : G_1^* \leq \Xi_2 \triangleright \varepsilon_2 t'_2 :: G_2^* : G_2^*$, or what is the same by the rule (\leq_{asc_ε}), we have to prove that $\varepsilon_1 \leq \varepsilon_2$, $\Omega \vdash \Xi_1 \triangleright t'_1 : G_1 \leq \Xi_2 \triangleright t'_2 : G_2$ and $G_1^* \sqsubseteq G_2^*$. By Proposition A.75, we know that $\varepsilon_1 = \mathcal{I}(\Xi, ()G_1, G_1^*) = \mathcal{I}(\Xi, ()G_1 \text{pullback} G_1^*, G_1 \text{pullback} G_1^*)$ and $\varepsilon_2 = \mathcal{I}(\Xi, ()G_2, G_2^*) = \mathcal{I}(\Xi, ()G_2 \text{pullback} G_2^*, G_2 \text{pullback} G_2^*)$. Since $G_1 \text{pullback} G_1^* \leq G_2 \text{pullback} G_2^*$, then $\varepsilon_1 = \mathcal{I}(\Xi, ()G_1, G_1^*) = \mathcal{I}(\Xi, ()G_1 \text{pullback} G_1^*, G_1 \text{pullback} G_1^*) \leq \mathcal{I}(\Xi, ()G_2 \text{pullback} G_2^*, G_2 \text{pullback} G_2^*) = \mathcal{I}(\Xi, ()G_2, G_2^*) = \varepsilon_2$, by Proposition A.77. Thus, we only have to prove that $\Omega \vdash \Xi_1 \triangleright t'_1 : G_1 \leq \Xi_2 \triangleright t'_2 : G_2$, and we know that $\Omega \vdash t'_1 : G_1 \leq_v t'_2 : G_2$, then by the induction hypothesis the result holds.

Case $(\Omega \vdash \Xi_1 \triangleright t_1 t'_1 : \text{cod}^\sharp(G_1) \leq \Xi_2 \triangleright t_2 t'_2 : \text{cod}^\sharp(G_2))$.

$$\begin{array}{c} \Omega \vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2 \quad \Omega \vdash \Xi_1 \triangleright t'_1 : G'_1 \leq \Xi_2 \triangleright t'_2 : G'_2 \\ \frac{G'_1 \text{pullback} \text{dom}^\sharp(G_1) \leq G'_2 \text{pullback} \text{dom}^\sharp(G_2)}{(\leq_{app}) \quad \Omega \vdash \Xi_1 \triangleright t_1 t'_1 : \text{cod}^\sharp(G_1) \leq \Xi_2 \triangleright t_2 t'_2 : \text{cod}^\sharp(G_2)} \\ \\ \frac{\Xi_1; \Delta; \Gamma_1 \vdash t_1 \rightsquigarrow t_{11} : G_1 \quad t'_{11} = \text{norm}(t_{11}, G_1, \text{dom}^\sharp(G_1) \rightarrow \text{cod}^\sharp(G_1)) \\ \Xi_1; \Delta; \Gamma_1 \vdash t'_1 \rightsquigarrow t_{12} : G'_1 \quad t'_{12} = \text{norm}(t_{12}, G'_1, \text{dom}^\sharp(G_1))}{(\text{Gapp}) \quad \Xi_1; \Delta; \Gamma_1 \vdash t_1 t'_1 \rightsquigarrow t'_{11} t'_{12} : \text{cod}^\sharp(G_1)} \\ \\ \frac{\Xi_2; \Delta; \Gamma_2 \vdash t_2 \rightsquigarrow t_{21} : G_2 \quad t'_{21} = \text{norm}(t_{21}, G_2, \text{dom}^\sharp(G_2) \rightarrow \text{cod}^\sharp(G_2)) \\ \Xi_2; \Delta; \Gamma_2 \vdash t'_2 \rightsquigarrow t_{22} : G'_2 \quad t'_{22} = \text{norm}(t_{22}, G'_2, \text{dom}^\sharp(G_2))}{(\text{Gapp}) \quad \Xi_2; \Delta; \Gamma_2 \vdash t_2 t'_2 \rightsquigarrow t'_{21} t'_{22} : \text{cod}^\sharp(G_2)} \end{array}$$

We have to prove that $\Omega \vdash \Xi_1 \triangleright t'_{11} t'_{12} : \text{cod}^\sharp(G_1) \leq \Xi_2 \triangleright t'_{21} t'_{22} : \text{cod}^\sharp(G_2)$, or what is the same by the rule (\leq_{app_ε}), we have to prove that $\Omega \vdash \Xi_1 \triangleright t'_{11} : \text{dom}^\sharp(G_1) \rightarrow \text{cod}^\sharp(G_1) \leq \Xi_2 \triangleright t'_{21} : \text{dom}^\sharp(G_2) \rightarrow \text{cod}^\sharp(G_2)$ and $\Omega \vdash \Xi_1 \triangleright t'_{12} : \text{dom}^\sharp(G_1) \leq \Xi_2 \triangleright t'_{22} : \text{dom}^\sharp(G_2)$. We know that

$$t'_{11} = \text{norm}(t_{11}, G_1, \text{dom}^\sharp(G_1) \rightarrow \text{cod}^\sharp(G_1)) = \varepsilon_{11} t_{11} :: \text{dom}^\sharp(G_1) \rightarrow \text{cod}^\sharp(G_1)$$

where $\varepsilon_{11} = \mathcal{I}(\Xi_1, ()G_1, \text{dom}^\sharp(G_1) \rightarrow \text{cod}^\sharp(G_1)) = \mathcal{I}(\Xi_1, ()\text{dom}^\sharp(G_1) \rightarrow \text{cod}^\sharp(G_1), \text{dom}^\sharp(G_1) \rightarrow \text{cod}^\sharp(G_1)) = \varepsilon_{\text{dom}^\sharp(G_1) \rightarrow \text{cod}^\sharp(G_1)}$

$$t'_{21} = \text{norm}(t_{21}, G_2, \text{dom}^\sharp(G_2) \rightarrow \text{cod}^\sharp(G_2)) = \varepsilon_{21} t_{21} :: \text{dom}^\sharp(G_2) \rightarrow \text{cod}^\sharp(G_2)$$

where $\varepsilon_{21} = \mathcal{I}(\Xi_2, ()G_2, \text{dom}^\sharp(G_2) \rightarrow \text{cod}^\sharp(G_2)) = \mathcal{I}(\Xi_2, ()\text{dom}^\sharp(G_2) \rightarrow \text{cod}^\sharp(G_2), \text{dom}^\sharp(G_2) \rightarrow \text{cod}^\sharp(G_2)) = \varepsilon_{\text{dom}^\sharp(G_2) \rightarrow \text{cod}^\sharp(G_2)}$

By induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$, we know that $\Omega \vdash \Xi_1 \triangleright t_{11} : G_1 \leq \Xi_2 \triangleright t_{21} : G_2$, and by Proposition A.76, we know that $G_1 \sqsubseteq G_2$, thus $\text{dom}^\sharp(G_1) \rightarrow \text{cod}^\sharp(G_1) \sqsubseteq \text{dom}^\sharp(G_2) \rightarrow \text{cod}^\sharp(G_2)$. Therefore, we only have to prove by rule (\leq_{Masc_ε}) that $\varepsilon_{11} \sqsubseteq \varepsilon_{21}$. But, by Proposition A.78 and $\text{dom}^\sharp(G_1) \rightarrow \text{cod}^\sharp(G_1) \sqsubseteq \text{dom}^\sharp(G_2) \rightarrow \text{cod}^\sharp(G_2)$ the results holds.

Also, we know that

$$t'_{12} = \text{norm}(t_{12}, G'_1, \text{dom}^\sharp(G_1)) = \varepsilon_{12} t_{12} :: \text{dom}^\sharp(G_1) \text{ where } \varepsilon_{12} = \mathcal{I}(\Xi_1, ()G'_1, \text{dom}^\sharp(G_1))$$

$$t'_{22} = \text{norm}(t_{22}, G'_2, \text{dom}^\sharp(G_2)) = \varepsilon_{22} t_{22} :: \text{dom}^\sharp(G_2) \text{ where } \varepsilon_{22} = \mathcal{I}(\Xi_2, ()G'_2, \text{dom}^\sharp(G_2))$$

By induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t'_1 : G'_1 \leq \Xi_2 \triangleright t'_2 : G'_2$, we know that $\Omega \vdash \Xi_1 \triangleright t_{12} : G'_1 \leq \Xi_2 \triangleright t_{22} : G'_2$. and and by Proposition A.76, we know that $dom^\sharp(G_1) \sqsubseteq dom^\sharp(G_2)$. By Proposition A.77 and $G'_1 pullback dom^\sharp(G_1) \leq G'_2 pullback dom^\sharp(G_2)$, we know that

$$\begin{aligned} \varepsilon_{12} &= \mathcal{J}(\Xi_1, ()G'_1, dom^\sharp(G_1)) = \mathcal{J}(\Xi_1, ()G'_1 pullback dom^\sharp(G_1), G'_1 pullback dom^\sharp(G_1)) \leq \\ &\mathcal{J}(\Xi_2, ()G'_2 pullback dom^\sharp(G_2), G'_2 pullback dom^\sharp(G_2)) = \mathcal{J}(\Xi_2, ()G'_2, dom^\sharp(G_2)) = \varepsilon_{22} \end{aligned}$$

Therefore, the results holds.

Case $(\Omega \vdash \Xi_1 \triangleright t_1 [G'_1] : inst^\sharp(G_1, G'_1) \leq \Xi_2 \triangleright t_2 [G'_2] : inst^\sharp(G_2, G'_2))$.

$$\begin{aligned} &(\leq_{appG}) \frac{\Omega \vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2 \quad G'_1 \leq G'_2}{\Omega \vdash \Xi_1 \triangleright t_1 [G'_1] : inst^\sharp(G_1, G'_1) \leq \Xi_2 \triangleright t_2 [G'_2] : inst^\sharp(G_2, G'_2)} \\ &(\text{GappG}) \frac{\Xi_1; \Delta; \Gamma_1 \vdash t_1 \rightsquigarrow t'_1 : G_1 \quad t'_1 = norm(t'_1, G_1, \forall var^\sharp(G_1).schm_u^\sharp(G_1))}{\Xi_1; \Delta; \Gamma_1 \vdash t_1 [G'_1] \rightsquigarrow t'_1 [G'_1] : inst^\sharp(G_1, G'_1)} \\ &(\text{GappG}) \frac{\Xi_2; \Delta; \Gamma_2 \vdash t_2 \rightsquigarrow t'_2 : G_2 \quad t'_2 = norm(t'_2, G_2, \forall var^\sharp(G_2).schm_u^\sharp(G_2))}{\Xi_2; \Delta; \Gamma_2 \vdash t_2 [G'_2] \rightsquigarrow t'_2 [G'_2] : inst^\sharp(G_2, G'_2)} \end{aligned}$$

We have to prove that $\Omega \vdash \Xi_1 \triangleright t'_1 [G'_1] : G_1^* \leq \Xi_2 \triangleright t'_2 [G'_2] : G_2^*$, or what is the same by the rule $(\leq_{appG_\varepsilon})$, we have to prove that $t'_1 \leq t'_2$ and $G'_1 \leq G'_2$. $G'_1 \leq G'_2$ follows by premise. We know that

$$t'_1 = norm(t'_1, G_1, \forall var^\sharp(G_1).schm_u^\sharp(G_1)) = \varepsilon_1 t'_1 :: \forall var^\sharp(G_1).schm_u^\sharp(G_1)$$

where $\varepsilon_1 = \mathcal{J}(\Xi_1, ()G_1, \forall var^\sharp(G_1).schm_u^\sharp(G_1)) = \mathcal{J}(\Xi_1, ()\forall var^\sharp(G_1).schm_u^\sharp(G_1), \forall var^\sharp(G_1).schm_u^\sharp(G_1)) = \varepsilon_{\forall var^\sharp(G_1).schm_u^\sharp(G_1)}$

$$t'_2 = norm(t'_2, G_2, \forall var^\sharp(G_2).schm_u^\sharp(G_2)) = \varepsilon_2 t'_2 :: \forall var^\sharp(G_2).schm_u^\sharp(G_2)$$

where $\varepsilon_2 = \mathcal{J}(\Xi_2, ()G_2, \forall var^\sharp(G_2).schm_u^\sharp(G_2)) = \mathcal{J}(\Xi_2, ()\forall var^\sharp(G_2).schm_u^\sharp(G_2), \forall var^\sharp(G_2).schm_u^\sharp(G_2)) = \varepsilon_{\forall var^\sharp(G_2).schm_u^\sharp(G_2)}$

By induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$, we know that $\Omega \vdash \Xi_1 \triangleright t'_1 : G_1 \leq \Xi_2 \triangleright t'_2 : G_2$, and by Proposition A.76, we know that $G_1 \sqsubseteq G_2$, thus $\forall var^\sharp(G_1).schm_u^\sharp(G_1) \sqsubseteq \forall var^\sharp(G_2).schm_u^\sharp(G_2)$. Therefore, we only have to prove by rule $(\leq_{Masc_\varepsilon})$ that $\varepsilon_1 \sqsubseteq \varepsilon_2$. But, by Proposition A.78 and $\forall var^\sharp(G_1).schm_u^\sharp(G_1) \sqsubseteq \forall var^\sharp(G_2).schm_u^\sharp(G_2)$ the results holds.

□

A.6 GSF: Parametricity

In this section we present the logical relation for parametricity of GSF, the proof of the fundamental property, and the soundness of the logical relation wrt contextual approximation.

A.6.1 Auxiliary Definitions

In this section we show function definitions used in the logical relation of GSF (Figure 3.15).

Definition A.81 $ev(\varepsilon u :: G) = \varepsilon$

Definition A.82

$$const(E) = \begin{cases} B & E = B \\ ? \rightarrow ? & E = E_1 \rightarrow E_2 \\ \forall X. ? & E = \forall X. E_1 \\ ? \times ? & E = E_1 \times E_2 \\ \alpha & E = \alpha^{E_1} \\ X & E = X \\ ? & E = ? \end{cases}$$

A.6.2 Fundamental Property

Theorem A.83 (Fundamental Property) *If $\Xi; \Delta; \Gamma \vdash t : G$ then $\Xi; \Delta; \Gamma \vdash t \preceq t : G$.*

PROOF. By induction on the type derivation of t .

Case (Easc). Then $t = \varepsilon s :: G$, and therefore:

$$(Easc) \frac{\Xi; \Delta; \Gamma \vdash s : G' \quad \varepsilon \Vdash \Xi; \Delta \vdash G' \sim G}{\Xi; \Delta; \Gamma \vdash \varepsilon s :: G : G}$$

We follow by induction on the structure of s .

- If $s = b$ then:

$$(Eb) \frac{ty(b) = B \quad \Xi; \Delta \vdash \Gamma}{\Xi; \Delta; \Gamma \vdash b : B}$$

Then we have to prove that $\Xi; \Delta; \Gamma \vdash \varepsilon b :: G \preceq \varepsilon b :: G : G$, but the result follows directly by Prop A.84 (Compatibility of Constant).

- If $s = \lambda x : G_1. t'$ then:

$$(E\lambda) \frac{\Xi; \Delta; \Gamma, x : G_1 \vdash t' : G_2}{\Xi; \Delta; \Gamma \vdash \lambda x : G_1. t' : G_1 \rightarrow G_2}$$

Then we have to prove that:

$$\Xi; \Delta; \Gamma \vdash \varepsilon(\lambda x : G_1. t') :: G \preceq \varepsilon(\lambda x : G_1. t') :: G : G$$

By induction hypotheses we already know that $\Xi; \Delta; \Gamma, x : G_1 \vdash t' \preceq t' : G_2$. But the result follows directly by Prop A.85 (Compatibility of term abstraction).

- If $s = \Lambda X.t'$ then:

$$(E\Lambda) \frac{\Xi; \Delta, X; \Gamma \vdash t' : G^* \quad \Xi; \Delta \vdash \Gamma}{\Xi; \Delta; \Gamma \vdash \Lambda X.t' : \forall X.G^*}$$

Then we have to prove that:

$$\Xi; \Delta; \Gamma \vdash \varepsilon(\Lambda X.t') :: G \preceq \varepsilon(\Lambda X.t') :: G : G$$

By induction hypotheses we already know that $\Xi; \Delta, X; \Gamma \vdash t' \preceq t' : G^*$. But the result follows directly by Prop 3.29 (Compatibility of type abstraction).

- If $s = \langle u_1, u_2 \rangle$ then:

$$(E\text{pair}) \frac{\Xi; \Delta; \Gamma \vdash u_1 : G_1 \quad \Xi; \Delta; \Gamma \vdash u_2 : G_2}{\Xi; \Delta; \Gamma \vdash \langle u_1, u_2 \rangle : G_1 \times G_2}$$

Then we have to prove that:

$$\Xi; \Delta; \Gamma \vdash \varepsilon\langle u_1, u_2 \rangle :: G \preceq \varepsilon\langle u_1, u_2 \rangle :: G : G$$

We know by premise that $\Xi; \Delta; \Gamma \vdash \pi_1(\varepsilon)u_1 :: G_1 : G_1$ and $\Xi; \Delta; \Gamma \vdash \pi_2(\varepsilon)u_2 :: G_2 : G_2$. Then by induction hypotheses we already know that: $\Xi; \Delta; \Gamma \vdash \pi_1(\varepsilon)u_1 :: G_1 \preceq \pi_1(\varepsilon)u_1 :: G_1 : G_1$ and $\Xi; \Delta; \Gamma \vdash \pi_2(\varepsilon)u_2 :: G_2 \preceq \pi_2(\varepsilon)u_2 :: G_2 : G_2$. But the result follows directly by Prop A.87 (Compatibility of pairs).

- If $s = t'$, and therefore:

$$(E\text{asc}) \frac{\Xi; \Delta; \Gamma \vdash t' : G' \quad \varepsilon \vdash \Xi; \Delta \vdash G' \sim G}{\Xi; \Delta; \Gamma \vdash \varepsilon t' :: G : G}$$

By induction hypotheses we already know that $\Xi; \Delta; \Gamma \vdash t' \preceq t' : G'$, then the result follows directly by Prop A.90 (Compatibility of ascriptions).

Case (Epair). Then $t = \langle t_1, t_2 \rangle$, and therefore:

$$(E\text{pair}) \frac{\Xi; \Delta; \Gamma \vdash t_1 : G_1 \quad \Xi; \Delta; \Gamma \vdash t_2 : G_2}{\Xi; \Delta; \Gamma \vdash \langle t_1, t_2 \rangle : G_1 \times G_2}$$

where $G = G_1 \times G_2$ Then we have to prove that:

$$\Xi; \Delta; \Gamma \vdash \langle t_1, t_2 \rangle \preceq \langle t_1, t_2 \rangle : G_1 \times G_2$$

By induction hypotheses we already know that: $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_1 : G_1$ and $\Xi; \Delta; \Gamma \vdash t_2 \preceq t_2 : G_2$. But the result follows directly by Prop A.88 (Compatibility of pairs).

Case (Ex). Then $t = x$, and therefore:

$$(E\text{x}) \frac{x : G \in \Gamma \quad \Xi; \Delta \vdash \Gamma}{\Xi; \Delta; \Gamma \vdash x : G}$$

Then we have to prove that $\Xi; \Delta; \Gamma \vdash x \preceq x : G$. But the result follows directly by Prop A.89 (Compatibility of variables).

Case (Eop). Then $t = op(\bar{t}')$, and therefore:

$$(Eop) \frac{\Xi; \Delta; \Gamma \vdash \bar{t}' : \bar{G}' \quad ty(op) = \bar{G}' \rightarrow G}{\Xi; \Delta; \Gamma \vdash op(\bar{t}') : G}$$

Then we have to prove that: $\Xi; \Delta; \Gamma \vdash op(\bar{t}') \preceq op(\bar{t}') : G$. By the induction hypothesis we obtain that: $\Xi; \Delta; \Gamma \vdash \bar{t}' \preceq \bar{t}' : \bar{G}$. Then the result follows directly by Prop A.91 (Compatibility of app operator).

Case (Eapp). Then $t = t_1 t_2$, and therefore:

$$(Eapp) \frac{\Xi; \Delta; \Gamma \vdash t_1 : G_{11} \rightarrow G_{12} \quad \Xi; \Delta; \Gamma \vdash t_2 : G_{11}}{\Xi; \Delta; \Gamma \vdash t_1 t_2 : G_{12}}$$

where $G = G_{12}$. Then we have to prove that:

$$\Xi; \Delta; \Gamma \vdash t_1 t_2 \preceq t_1 t_2 : G_{12}$$

By the induction hypothesis we obtain that: $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_1 : G_{11} \rightarrow G_{12}$ and $\Xi; \Delta; \Gamma \vdash t_2 \preceq t_2 : G_{11}$. Then the result follows directly by Prop A.92 (Compatibility of term application).

Case (EappG). Then $t = t' [G_2]$, and therefore:

$$(EappG) \frac{\Xi; \Delta; \Gamma \vdash t' : \forall X. G_1 \quad \Xi; \Delta \vdash G_2}{\Xi; \Delta; \Gamma \vdash t' [G_2] : G_1[G_2/X]}$$

where $G = G_1[G_2/X]$. Then we have to prove that:

$$\Xi; \Delta; \Gamma \vdash t' [G_2] \preceq t' [G_2] : G_1[G_2/X]$$

By induction hypotheses we know that:

$$\Xi; \Delta; \Gamma \vdash t' \preceq t' : \forall X. G_1$$

Then the result follows directly by Prop 3.30 (Compatibility of type application).

Case (Epair1). Then $t = \pi_1(t')$, and therefore:

$$(Epair1) \frac{\Xi; \Delta; \Gamma \vdash t' : G_1 \times G_2}{\Xi; \Delta; \Gamma \vdash \pi_1(t') : G_1}$$

where $G = G_1$. Then we have to prove that: $\Xi; \Delta; \Gamma \vdash \pi_1(t') \preceq \pi_1(t') : G_1$. By the induction hypothesis we obtain that: $\Xi; \Delta; \Gamma \vdash t' \preceq t' : G_1 \times G_2$. Then the result follows directly by Prop A.94 (Compatibility of access to the first component of the pair).

Case (Epair2). Then $t = \pi_2(t')$, and therefore:

$$(Epair2) \frac{\Xi; \Delta; \Gamma \vdash t' : G_1 \times G_2}{\Xi; \Delta; \Gamma \vdash \pi_2(t') : G_2}$$

where $G = G_2$. Then we have to prove that: $\Xi; \Delta; \Gamma \vdash \pi_2(t') \preceq \pi_2(t') : G_2$. By the induction hypothesis we obtain that: $\Xi; \Delta; \Gamma \vdash t' \preceq t' : G_1 \times G_2$. Then the result follows directly by Prop A.95 (Compatibility of access to the second component of the pair).

□

In order to prove parametricity, we add an index to the evidence and we are more detailed in the reduction rules. A brief explanation is given below. The index of an evidence is an integer greater than zero. To know the index of an evidence ε , we use the following operator $\varepsilon.n = k$, which specifies that the index of the evidence ε is the integer $k > 0$. The reduction rules always took a step. Here we redefine them and they can take one or more steps. This will depend on whether or not a transitivity of evidence is applied. If it does, the rule will take as many steps as the evidence index on the right. Below we define the steps in the rules

$\Xi \triangleright t \longrightarrow \Xi \triangleright t$ or error		Notion of reduction	
$(Rasc)$	$\Xi \triangleright \varepsilon_2(\varepsilon_1 u :: G_1) :: G_2$	\xrightarrow{k}	$\begin{cases} \Xi \triangleright (\varepsilon_1 \circledast \varepsilon_2)u :: G_2 & \text{if } \varepsilon_2.n = k \\ \mathbf{error} & \text{if not defined} \end{cases}$
(Rop)	$\Xi \triangleright op(\overline{\varepsilon u} :: \overline{G})$	$\xrightarrow{1}$	$\Xi \triangleright \varepsilon_B \delta(op, \bar{u}) :: B$ where $B \triangleq cod(ty(op))$
$(Rapp\vec{\Xi})$	$\Xi \triangleright (\varepsilon_1(\lambda x : G_{11}.t) :: G_1 \rightarrow G_2) (\varepsilon_2 u :: G_1)$	$\xrightarrow{k+1}$	$\begin{cases} \Xi \triangleright cod(\varepsilon_1)(t[(\varepsilon_2 \circledast dom(\varepsilon_1))u :: G_{11}]/x) :: G_2 \\ \text{if } dom(\varepsilon_1) = k \\ \mathbf{error} & \text{if not defined} \end{cases}$
$(Rpair)$	$\Xi \triangleright \langle \varepsilon_1 u_1 :: G_1, \varepsilon_2 u_2 :: G_2 \rangle$	$\xrightarrow{1}$	$\Xi \triangleright (\varepsilon_1 \times \varepsilon_2)\langle u_1, u_2 \rangle :: G_1 \times G_2$
$(Rproj_i)$	$\Xi \triangleright \pi_i(\varepsilon \langle u_1, u_2 \rangle :: G_1 \times G_2)$	$\xrightarrow{1}$	$\Xi \triangleright p_i(\varepsilon)u_i :: G_i$
$(RappG)$	$\Xi \triangleright (\varepsilon \Lambda X.t :: \forall X.G) [G']$	$\xrightarrow{1}$	$\Xi' \triangleright \varepsilon_{out}(\varepsilon[\hat{\alpha}]t[\hat{\alpha}/X] :: G[\alpha/X]) :: G[G'/X]$ where $\Xi' \triangleq \Xi, \alpha := G'$ for some $\alpha \notin dom(\Xi)$ and $\hat{\alpha} = lift_{\Xi'}(\alpha)$

Proposition A.84 (Compatibility-Eb) *If $b \in B$, $\varepsilon \vdash \Xi; \Delta \vdash B \sim G$ and $\Xi; \Delta \vdash \Gamma$ then:*

$$\Xi; \Delta; \Gamma \vdash \varepsilon b :: G \preceq \varepsilon b :: G : G$$

PROOF. As b is constant then it does not have free variables or type variables, then $b = \rho(\gamma_i(b))$. Then we have to prove that for all $W \in \mathcal{S}[\Xi]$ it is true that:

$$(W, \rho_1(\varepsilon)b :: \rho(G), \rho_2(\varepsilon)b :: \rho(G)) \in \mathcal{T}_\rho[G]$$

As $\rho_i(\varepsilon)b :: G$ are values, then we have to prove that:

$$(W, \rho_1(\varepsilon)b :: \rho(G), \rho_2(\varepsilon)b :: \rho(G)) \in \mathcal{V}_\rho[G]$$

1. $G = B$, we know that $\langle B, B \rangle = \varepsilon \vdash \Xi; \Delta \vdash B \sim B$, then $\rho_i(\varepsilon) = \varepsilon$ and the result follows immediately by the definition of $\mathcal{V}_\rho[B]$.
2. If $G \in \text{TYPE_NAME}$ then $\varepsilon = \langle H_3, \alpha^{E_4} \rangle$. Notice that as α^{E_4} cannot have free type variables therefore H_3 neither. Then $\varepsilon = \rho_i(\varepsilon)$. As α is sync, then let us call $G'' = W.\Xi_i(\alpha)$. We have to prove that:

$$(W, \langle H_3, \alpha^{E_4} \rangle b :: \alpha, \langle H_3, \alpha^{E_4} \rangle b :: \alpha) \in \mathcal{V}_\rho[\alpha]$$

which, by definition of $\mathcal{V}_\rho[\alpha]$, is equivalent to prove that:

$$(\downarrow W, \langle H_3, E_4 \rangle b :: G'', \langle E_3, E_4 \rangle b :: G'') \in \mathcal{V}_\rho[G'']$$

Then we proceed by case analysis on ε :

- (Case $\varepsilon = \langle H_3, \alpha^{\beta^{E_4}} \rangle$). We know that $\langle H_3, \alpha^{\beta^{E_4}} \rangle \vdash \Xi; \Delta \vdash B \sim \alpha$, then by Lemma A.114, $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash B \sim G''$. As $\beta^{E_4} \sqsubseteq G''$, then G'' can either be $?$ or β .

If $G'' = ?$, then by definition of $\mathcal{V}_\rho[[?]]$, we have to prove that the resulting values belong to $\mathcal{V}_\rho[[\beta]]$. Also as $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash B \sim ?$, by Lemma A.112, $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash B \sim \beta$, and then we proceed just like this case once again (this is process is finite as there are no circular references by construction and it ends up in something different to a type name). If $G'' = \beta$ we use an analogous argument as for $G'' = ?$.

- (Case $\varepsilon = \langle H_3, \alpha^{H_4} \rangle$). We have to prove that

$$(\downarrow W, \langle H_3, H_4 \rangle b :: G'', \langle H_3, H_4 \rangle b :: G'') \in \mathcal{V}_\rho[[G'']]$$

By Lemma A.114, $\langle H_3, H_4 \rangle \vdash \Xi; \Delta \vdash B \sim G''$. Then if $G'' = ?$, we proceed as the case $G = ?$, with the evidence $\varepsilon = \langle H_3, H_4 \rangle$. If $G'' \in \text{HEADTYPE}$, we proceed as the previous case where $G = B$, and the evidence $\varepsilon = \langle H_3, H_4 \rangle$.

Also, we have to prove that $(\forall \Xi', \varepsilon', G_1^*$, such that $\varepsilon'.n = k$, $\varepsilon' = \langle \alpha^{E_1^{**}}, E_2^{**} \rangle$ ($\downarrow W \in \mathcal{S}[[\Xi']] \wedge \varepsilon' \vdash \Xi' \vdash \alpha \sim G_1^*$), we get that

$$(\downarrow_1 W, \varepsilon'(\langle H_3, \alpha^{H_4} \rangle u_1 :: \alpha) :: G_1^*, \varepsilon'(\langle H_4, \alpha^{E_{22}} \rangle u_2 :: \alpha) :: G_1^*) \in \mathcal{F}_\rho[[G_1^*]]$$

or what is the same ($(\langle H_3, \alpha^{H_4} \rangle \circ \varepsilon')$ fails the result follows immediately)

$$(\downarrow_{1+k} W, (\langle H_3, \alpha^{H_4} \rangle \circ \varepsilon') u_1 :: G_1^*, (\langle H_2, \alpha^{H_4} \rangle \circ \varepsilon') u_2 :: G_1^*) \in \mathcal{V}_\rho[[G_1^*]]$$

By definition of transitivity and Lemma A.115, we know that

$$\langle H_3, \alpha^{H_4} \rangle \circ \langle \alpha^{E_1^{**}}, E_2^{**} \rangle = \langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle$$

We know that $\langle E_1^{**}, E_2^{**} \rangle \vdash \Xi' \vdash G'' \sim G_1^*$. Since $\langle E_1^{**}, E_2^{**} \rangle \vdash \Xi \vdash G'' \sim G_1^*$, $\downarrow_1 W \in \mathcal{S}[[\Xi']]$, $(\downarrow_1 W, \langle H_3, H_4 \rangle u_1 :: G'', \langle H_1, H_4 \rangle u_2 :: G'') \in \mathcal{V}_\rho[[G'']]$, by Lemma A.101, we know that (since $(\langle H_3, \alpha^{H_4} \rangle \circ \varepsilon')$ does not fail then $(\langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle)$ also does not fail by the transitivity rules)

$$(\downarrow_{1+k} W, (\langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle) u_1 :: G_1^*, (\langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle) u_2 :: G_1^*) \in \mathcal{V}_\rho[[G_1^*]]$$

The result follows immediately.

3. If $G = ?$ we have the following cases:

- ($G = ?, \varepsilon = \langle H_3, H_4 \rangle$). By the definition of $\mathcal{V}_\rho[[?]]$ in this case we have to prove that:

$$(W, \rho_1(\varepsilon)b :: \text{const}(H_4), \rho_2(\varepsilon)b :: \text{const}(H_4)) \in \mathcal{V}_\rho[[\text{const}(H_4)]]$$

but as $\text{const}(H_4) = B$ (note that $H_3 = B$ then since $H_4 \in \text{HEADTYPE}$ has to be B). The the result follows immediately since is part of the premise.

- ($G = ?, \varepsilon = \langle H_3, \alpha^{E_4} \rangle$). Notice that as α^{E_4} cannot have free type variables therefore E_3 neither. Then $\varepsilon = \rho_i(\varepsilon)$. By the definition of $\mathcal{V}_\rho[[?]]$ we have to prove that

$$(W, \langle H_3, \alpha^{E_4} \rangle u_1 :: \alpha, \langle H_3, \alpha^{E_4} \rangle u_2 :: \alpha) \in \mathcal{V}_\rho[[\alpha]]$$

Note that by Lemma A.112 we know that $\varepsilon \vdash \Xi; \Delta \vdash B \sim \alpha$. Then we proceed just like the case $G \in \text{TYPENAME}$.

□

Proposition A.85 (Compatibility-E λ) *If $\Xi; \Delta; \Gamma, x : G_1 \vdash t \preceq t' : G_2$, $\varepsilon \vdash \Xi; \Delta \vdash G_1 \rightarrow G_2 \sim G$ then:*

$$\Xi; \Delta; \Gamma \vdash \varepsilon(\lambda x : G_1.t) :: G \preceq \varepsilon(\lambda x : G_1.t') :: G : G$$

PROOF. First, we are required to show that $\Xi; \Delta; \Gamma \vdash \varepsilon(\lambda x : G_1.t) :: G : G$ and $\Xi; \Delta; \Gamma \vdash \varepsilon(\lambda x : G_1.t') :: G : G$, which follow from $\varepsilon \vdash \Xi; \Delta \vdash G_1 \rightarrow G_2 \sim G$ and $\Xi; \Delta; \Gamma \vdash \lambda x : G_1.t : G_1 \rightarrow G_2$ and $\Xi; \Delta; \Gamma \vdash \lambda x : G_1.t' : G_1 \rightarrow G_2$ respectively, which follow (respectively) from $\Xi; \Delta; \Gamma, x : G_1 \vdash t : G_2$ and $\Xi; \Delta; \Gamma, x : G_1 \vdash t' : G_2$, which follow from $\Xi; \Delta; \Gamma, x : G_1 \vdash t \preceq t' : G_2$.

Consider arbitrary W, ρ, γ such that $W \in \mathcal{S}[\Xi]$, $(W, \rho) \in \mathcal{D}[\Delta]$ and $(W, \gamma) \in \mathcal{G}_\rho[\Gamma]$. We are required to show that:

$$(W, \rho(\gamma_1(\varepsilon(\lambda x : G_1.t) :: G)), \rho(\gamma_2(\varepsilon(\lambda x : G_1.t) :: G))) \in \mathcal{T}_\rho[G]$$

Consider arbitrary i, v_1 and Ξ_1 such that $i < W.j$ and:

$$W.\Xi_1 \triangleright \rho(\gamma_1(\varepsilon(\lambda x : G_1.t) :: G)) \longrightarrow^i \Xi_1 \triangleright v_1$$

Since $\rho(\gamma_1(\varepsilon(\lambda x : G_1.t) :: G)) = \varepsilon_1^\rho(\lambda x : \rho(G_1).\rho(\gamma_1(t))) :: \rho(G)$ and $\varepsilon_2^\rho(\lambda x : \rho(G_1).\rho(\gamma_2(t))) :: \rho(G)$ is already a value, where $\varepsilon_i^\rho = \rho_i(\varepsilon)$, we have $i = 0$ and $v_1 = \varepsilon_1^\rho(\lambda x : \rho(G_1).\rho(\gamma_1(t))) :: \rho(G)$ and $\Xi_1 = W.\Xi_1$. Since $\varepsilon_2^\rho(\lambda x : \rho(G_1).\rho(\gamma_2(t))) :: \rho(G)$ is already a value, we are required to show that $\exists W'$, such that $W'.j + i = W.j$, $W' \succeq W$, $W'.\Xi_1 = \Xi_1$, $W'.\Xi_2 = \Xi_2$ and:

$$(W', \varepsilon_1^\rho(\lambda x : \rho(G_1).\rho(\gamma_1(t))) :: \rho(G), \varepsilon_2^\rho(\lambda x : \rho(G_1).\rho(\gamma_2(t))) :: \rho(G)) \in \mathcal{V}_\rho[G]$$

Let $W' = W$, then we have to show that:

$$(W, \varepsilon_1^\rho(\lambda x : \rho(G_1).\rho(\gamma_1(t))) :: \rho(G), \varepsilon_2^\rho(\lambda x : \rho(G_1).\rho(\gamma_2(t))) :: \rho(G)) \in \mathcal{V}_\rho[G]$$

Let's suppose that $\varepsilon_1^\rho.n = k$.

First we have to prove that:

$$W.\Xi_1; \Delta; \Gamma \vdash \varepsilon_1^\rho(\lambda x : \rho(G_1).\rho(\gamma_1(t))) :: \rho(G) : \rho(G)$$

As we know that $\Xi; \Delta; \Gamma \vdash \varepsilon(\lambda x : G_1.t) :: G : G$, by Lemma A.110 the result follows immediately. The case $W.\Xi_2; \Delta; \Gamma \vdash \varepsilon_2^\rho(\lambda x : \rho(G_1).\rho(\gamma_2(t))) :: \rho(G) : \rho(G)$ is similar.

The type G can be $G'_1 \rightarrow G'_2$, for some G'_1 and G'_2 , or ? or a TYPENAME.

1. $G = G'_1 \rightarrow G'_2$, we are required to show that $\forall W'', v'_1 = \varepsilon'_1 u'_1 :: \rho(G'_1), v'_2 = \varepsilon'_2 u'_2 :: \rho(G'_1)$, such that $W'' \succeq W$ and $(\downarrow W'', v'_1, v'_2) \in \mathcal{V}_\rho[G'_1]$, it is true that:

$$(W'', \varepsilon_1^\rho(\lambda x : \rho(G_1).\rho(\gamma_1(t))) :: \rho(G'_1 \rightarrow G'_2) v'_1, \varepsilon_2^\rho(\lambda x : \rho(G_1).\rho(\gamma_2(t))) :: \rho(G'_1 \rightarrow G'_2) v'_2) \in \mathcal{T}_\rho[G'_2]$$

If $(\varepsilon'_1 \# \text{dom}(\varepsilon_1^\rho))$ fails, then by Lemma A.111 $(\varepsilon'_2 \# \text{dom}(\varepsilon_2^\rho))$ and the result follows immediately.

Else, if $(\varepsilon'_i \# \text{dom}(\varepsilon_i^\rho))$ follows, where $\text{dom}(\varepsilon_1^\rho).n = k$, we know that

$$W''.\Xi_1 \triangleright \varepsilon_1^\rho(\lambda x : \rho(G_1).\rho(\gamma_1(t))) :: \rho(G'_1 \rightarrow G'_2) v'_1 \longrightarrow^{k+1}$$

$$\begin{aligned} W''.\Xi_1 \triangleright \text{cod}(\varepsilon_1^\rho)(\rho(\gamma_1(t))[(\varepsilon'_1 \circ \text{dom}(\varepsilon_1^\rho))u'_1 :: \rho(G_1)]/x] :: \rho(G'_2) v'_1 \longrightarrow^{k^*} \\ \Xi_1 \triangleright \text{cod}(\varepsilon_1^\rho)v_{1f} :: \rho(G'_2) \longrightarrow k \\ \Xi_1 \triangleright v_1^* \end{aligned}$$

Thus, we have to prove that there exists W^* , such that:

$$W''.\Xi_2 \triangleright \varepsilon_2^\rho(\lambda x : \rho(G_1).\rho(\gamma_2(t'))) :: \rho(G'_1 \rightarrow G'_2) v'_2 \longrightarrow^* \Xi_2 \triangleright v_2^*$$

and $(W^*, v_1^*, v_2^*) \in \mathcal{V}_\rho[[G'_2]]$, $W^*.j + 1 + 2k + k^* = W''.j$, $W^*.\Xi_1 = \Xi_1$ and $W^*.\Xi_2 = \Xi_2$.

Note that $\text{dom}(\varepsilon_i^\rho) \vdash W''.\Xi_i \vdash \rho(G'_1) \sim \rho(G_1)$. By the Lemma A.101 (with the type G_1 and the evidences $\text{dom}(\varepsilon_i^\rho) \vdash W''.\Xi_i \vdash \rho(G'_1) \sim \rho(G_1)$) it is true that:

$$(\downarrow_1 W'', \text{dom}(\varepsilon_1^\rho)v'_1 :: G_1, \text{dom}(\varepsilon_2^\rho)v'_2 :: G_1) \in \mathcal{J}_\rho[[G_1]]$$

Since $(\varepsilon'_i \circ \text{dom}(\varepsilon_i^\rho))$ does not fail, it is true that:

$$((\downarrow_{k+1} W''), (\varepsilon'_1 \circ \text{dom}(\varepsilon_1^\rho))u'_1 :: G_1, (\varepsilon'_2 \circ \text{dom}(\varepsilon_2^\rho))u'_2 :: G_1) \in \mathcal{V}_\rho[[G_1]]$$

We instantiate the hypothesis $\Xi; \Delta; \Gamma \vdash t \preceq t' : G_2$, with $(\downarrow_{k+1} W'')$, ρ and $\gamma[x : \rho(G_1) \mapsto (v''_1, v''_2)]$, where $v''_i = (\varepsilon'_i \circ \text{dom}(\varepsilon_i^\rho))u'_i :: \rho(G_1)$. Note that $\mathcal{S}[\Xi] \ni (\downarrow_{k+1} W'') \succeq W$ by the definition of $\mathcal{S}[\Xi]$, $((\downarrow_{k+1} W''), \rho) \in \mathcal{D}[\Delta]$ by the definition of $\mathcal{D}[\Delta]$ and $((\downarrow_{k+1} W''), \gamma[x \mapsto (v''_1, v''_2)]) \in \mathcal{G}_\rho[[\Gamma, x : \rho(G_1)]]$, which follow from: $((\downarrow_{k+1} W''), \gamma) \in \mathcal{G}_\rho[[\Gamma]]$ and $((\downarrow_{k+1} W''), v''_1, v''_2) \in \mathcal{V}_\rho[[G_1]]$ which follows from above. Then, we have that:

$$((\downarrow_{k+1} W''), \rho(\gamma_1(t))[v''_1/x], \rho(\gamma_2(t'))[v''_2/x]) \in \mathcal{J}_\rho[[G_2]]$$

If the following term reduces to error, then the result follows immediately.

$$W''.\Xi_1 \triangleright \rho(\gamma_1(t))[v''_1/x]$$

If the above is not true, then the following terms reduce to values (v_{if}) and $\exists W''' \succeq (\downarrow_{k+1} W'')$ such that $(W''', v_{1f}, v_{2f}) \in \mathcal{V}_\rho[[G'_2]]$ and $W'''.j + k^* = (\downarrow_{k+1} W'').j$, or what is the same $W'''.j + k^* + k + 1 = (W'').j$.

$$W''.\Xi_1 \triangleright \rho(\gamma_1(t))[v''_1/x] \longrightarrow^{k^*} W'''.\Xi_1 \triangleright v_{1f}$$

$$W''.\Xi_2 \triangleright \rho(\gamma_2(t'))[v''_2/x] \longrightarrow^* W'''.\Xi_2 \triangleright v_{2f}$$

We instantiate the induction hypothesis in the previous result $((W''', v_{1f}, v_{2f}))$ with the type G'_2 and the evidence $\text{cod}(\varepsilon_i^\rho) \vdash W'''.\Xi_i \vdash G'_2 \sim G'_2$, then we obtain that:

$$(W''', \text{cod}(\varepsilon_1^\rho)v_{1f} :: \rho(G'_2), \text{cod}(\varepsilon_2^\rho)v_{2f} :: \rho(G'_2)) \in \mathcal{J}_\rho[[G'_2]]$$

Therefore, we get $(\downarrow_k W''', v_1^*, v_2^*) \in \mathcal{V}_\rho[[G'_2]]$. Taking $W^* = (\downarrow_k W''')$, the result follows immediately. Note that $W'''.j + k + k^* + 1 = W'''.j$ and therefore $(\downarrow_k W''').j + 1 + 2k + k^* = W'''.j$.

For the other cases of G , let's consider that $u_1 = \lambda x : \rho(G_1).\rho(\gamma_1(t))$, $u_2 = \lambda x : \rho(G_1).\rho(\gamma_2(t'))$ and $G^* = G_1 \rightarrow G_2$, we have to prove that:

$$(W, \rho_1(\varepsilon)u_1 :: \rho(G), \rho_2(\varepsilon)u_2 :: \rho(G)) \in \mathcal{V}_\rho[[G]]$$

2. If $G \in \text{TYPE NAME}$ then $\varepsilon = \langle H_3, \alpha^{E_4} \rangle$. Notice that as α^{E_4} cannot have free type variables therefore H_3 neither. Then $\varepsilon = \rho_i(\varepsilon)$. As α is sync, then let us call $G'' = W.\Xi_i(\alpha)$. We have to prove that:

$$(W, \langle H_3, \alpha^{E_4} \rangle_{u_1} :: \alpha, \langle H_3, \alpha^{E_4} \rangle_{u_2} :: \alpha) \in \mathcal{V}_\rho[\alpha]$$

which, by definition of $\mathcal{V}_\rho[\alpha]$, is equivalent to prove that:

$$(\downarrow W, \langle H_3, E_4 \rangle_{u_1} :: G'', \langle E_3, E_4 \rangle_{u_2} :: G'') \in \mathcal{V}_\rho[G'']$$

Then we proceed by case analysis on ε :

- (Case $\varepsilon = \langle H_3, \alpha^{\beta^{E_4}} \rangle$). We know that $\langle H_3, \alpha^{\beta^{E_4}} \rangle \vdash \Xi; \Delta \vdash G^* \sim \alpha$, then by Lemma A.114, $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash G^* \sim G''$. As $\beta^{E_4} \sqsubseteq G''$, then G'' can either be $?$ or β .

If $G'' = ?$, then by definition of $\mathcal{V}_\rho[?]$, we have to prove that the resulting values belong to $\mathcal{V}_\rho[\beta]$. Also as $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash G^* \sim ?$, by Lemma A.112, $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash G^* \sim \beta$, and then we proceed just like this case once again (this is process is finite as there are no circular references by construction and it ends up in something different to a type name). If $G'' = \beta$ we use an analogous argument as for $G'' = ?$.

- (Case $\varepsilon = \langle H_3, \alpha^{H_4} \rangle$). We have to prove that

$$(\downarrow W, \langle H_3, H_4 \rangle_{u_1} :: G'', \langle H_3, H_4 \rangle_{u_2} :: G'') \in \mathcal{V}_\rho[G'']$$

By Lemma A.114, $\langle H_3, H_4 \rangle \vdash \Xi; \Delta \vdash G^* \sim G''$. Then if $G'' = ?$, we proceed as the case $G = ?$, with the evidence $\varepsilon = \langle H_3, H_4 \rangle$. If $G'' \in \text{HEAD TYPE}$, we proceed as the previous case where $G = G'_1 \rightarrow G'_2$, and the evidence $\varepsilon = \langle H_3, H_4 \rangle$.

Also, we have to prove that $(\forall \Xi', \varepsilon', G_1^*$, such that $\varepsilon'.n = k$, $\varepsilon' = \langle \alpha^{E_1^{**}}, E_2^{**} \rangle$ ($\downarrow W \in \mathcal{S}[\Xi'] \wedge \varepsilon' \vdash \Xi' \vdash \alpha \sim G_1^*$), we get that

$$(\downarrow_1 W, \varepsilon'(\langle H_3, \alpha^{H_4} \rangle_{u_1} :: \alpha) :: G_1^*, \varepsilon'(\langle H_4, \alpha^{E_{22}} \rangle_{u_2} :: \alpha) :: G_1^*) \in \mathcal{F}_\rho[G_1^*]$$

or what is the same ($(\langle H_3, \alpha^{H_4} \rangle \circ \varepsilon')$ fails the result follows immediately)

$$(\downarrow_{1+k} W, (\langle H_3, \alpha^{H_4} \rangle \circ \varepsilon')_{u_1} :: G_1^*, (\langle H_2, \alpha^{H_4} \rangle \circ \varepsilon')_{u_2} :: G_1^*) \in \mathcal{V}_\rho[G_1^*]$$

By definition of transitivity and Lemma A.115, we know that

$$\langle H_3, \alpha^{H_4} \rangle \circ \langle \alpha^{E_1^{**}}, E_2^{**} \rangle = \langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle$$

We know that $\langle E_1^{**}, E_2^{**} \rangle \vdash \Xi' \vdash G'' \sim G_1^*$. Since $\langle E_1^{**}, E_2^{**} \rangle \vdash \Xi \vdash G'' \sim G_1^*$, $\downarrow_1 W \in \mathcal{S}[\Xi']$, $(\downarrow_1 W, \langle H_3, H_4 \rangle_{u_1} :: G'', \langle H_1, H_4 \rangle_{u_2} :: G'') \in \mathcal{V}_\rho[G'']$, by Lemma A.101, we know that (since $(\langle H_3, \alpha^{H_4} \rangle \circ \varepsilon')$ does not fail then $(\langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle)$ also does not fail by the transitivity rules)

$$(\downarrow_{1+k} W, (\langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle)_{u_1} :: G_1^*, (\langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle)_{u_2} :: G_1^*) \in \mathcal{V}_\rho[G_1^*]$$

The result follows immediately.

3. If $G = ?$ we have the following cases:

- ($G = ?, \varepsilon = \langle H_3, H_4 \rangle$). By the definition of $\mathcal{V}_\rho[?]$ in this case we have to prove that:

$$(W, \rho_1(\varepsilon)_{u_1} :: \rho(G), \rho_2(\varepsilon)_{u_2} :: \rho(G)) \in \mathcal{V}_\rho[\text{const}(H_4)]$$

but as $\text{const}(H_4) = ? \rightarrow ?$, we proceed just like this case where $G = G'_1 \rightarrow G_2$, where $G'_1 = ?$ and $G'_2 = ?$.

- ($G = ?, \varepsilon = \langle H_3, \alpha^{E_4} \rangle$). Notice that as α^{E_4} cannot have free type variables therefore E_3 neither. Then $\varepsilon = \rho_i(\varepsilon)$. By the definition of $\mathcal{V}_\rho[\![?]\!]$ we have to prove that

$$(W, \langle H_3, \alpha^{E_4} \rangle u_1 :: \alpha, \langle H_3, \alpha^{E_4} \rangle u_2 :: \alpha) \in \mathcal{V}_\rho[\![\alpha]\!]$$

Note that by Lemma A.112 we know that $\varepsilon \vdash \Xi; \Delta \vdash G^* \sim \alpha$. Then we proceed just like the case $G \in \text{TYPE_NAME}$.

□

Lemma A.86 (Compatibility-E Λ) *If $\Xi; \Delta, X \vdash t_1 \preceq t_2 : G, \varepsilon \vdash \Xi; \Delta \vdash \forall X.G \sim G'$ and $\Xi; \Delta \vdash \Gamma$ then $\Xi; \Delta; \Gamma \vdash \varepsilon(\Lambda X.t_1) :: G' \preceq \varepsilon(\Lambda X.t_2) :: G' : G'$.*

PROOF. First, we are required to prove that $\Xi; \Delta; \Gamma \vdash \varepsilon(\Lambda X.t_i) :: G' : G'$, but by unfolding the premises we know that $\Xi; \Delta, X \vdash t_i : G$, therefore:

$$\frac{\Xi; \Delta, X; \Gamma \vdash t_i : G \quad \Xi; \Delta \vdash \Gamma}{\Xi; \Delta; \Gamma \vdash \Lambda X.t_i \in \forall X.G}$$

Then we can conclude that:

$$\frac{\Xi; \Delta; \Gamma \vdash \Lambda X.t_i \in \forall X.G \quad \varepsilon \vdash \Xi; \Delta \vdash \forall X.G \sim G'}{\Xi; \Delta; \Gamma \vdash \varepsilon(\Lambda X.t_i) :: G' : G'}$$

Consider arbitrary W, ρ, γ such that $W \in \mathcal{S}[\![\Xi]\!]$, $(W, \rho) \in \mathcal{D}[\![\Delta]\!]$ and $(W, \gamma) \in \mathcal{G}_\rho[\![\Gamma]\!]$. We are required to show that:

$$(W, \rho(\gamma_1(\varepsilon(\Lambda X.t_1) :: G')), \rho(\gamma_2(\varepsilon(\Lambda X.t_2) :: G'))) \in \mathcal{T}_\rho[\![G']\!]$$

First we have to prove that:

$$W.\Xi_i \vdash \rho(\gamma_i(\varepsilon(\Lambda X.t_i) :: G')) : \rho(G')$$

As we know that $\Xi; \Delta; \Gamma \vdash \varepsilon(\Lambda X.t_i) :: G' : G'$, by Lemma A.110 the result follows immediately.

By definition of substitutions $\rho(\gamma_i(\varepsilon(\Lambda X.t_i) :: G')) = \varepsilon_i^\rho(\Lambda X.\rho(\gamma_i(t_i))) :: \rho(G')$, where $\varepsilon_i^\rho = \rho_i(\varepsilon)$, therefore we have to prove that:

$$(W, \varepsilon_1^\rho(\Lambda X.\rho(\gamma_1(t_1))) :: \rho(G'), \varepsilon_2^\rho(\Lambda X.\rho(\gamma_2(t_2))) :: \rho(G')) \in \mathcal{T}_\rho[\![G']\!]$$

We already know that both terms are values and therefore we only have to prove that:

$$(W, \varepsilon_1^\rho(\Lambda X.\rho(\gamma_1(t_1))) :: \rho(G'), \varepsilon_2^\rho(\Lambda X.\rho(\gamma_2(t_2))) :: \rho(G')) \in \mathcal{V}_\rho[\![G']\!]$$

Let's suppose that $\varepsilon_1^\rho.n = k$.

The type G' can be $\forall X.G'_1$, for some $G'_1, ?$ or a `TYPE_NAME`. Let $u_1 = \Lambda X.\rho(\gamma_1(t_1))$, $u_2 = \Lambda X.\rho(\gamma_2(t_2))$ and $G^* = \forall X.G$, we have to prove that:

$$(W, \rho_1(\varepsilon)u_1 :: \rho(G), \rho_2(\varepsilon)u_2 :: \rho(G)) \in \mathcal{V}_\rho[\![G']\!]$$

1. If $G' = \forall X.G'_1$, then consider $W' \succeq W$, and G_1, G_2, R and α , such that $W'.\Xi_i \vdash G_i$, and $R \in \text{REL}_{W'.j}[G_1, G_2]$.

$$W'.\Xi_i \triangleright \varepsilon_i^\rho(\Lambda X.\rho(\gamma_i(t_i))) :: \forall X.\rho(G'_1) [G_i] \longrightarrow$$

$$W'.\Xi_i, \alpha := G_i \triangleright \varepsilon_{\forall X.\rho(G'_1)}^{E_i/\alpha^{E_i}}(\varepsilon_i^\rho[\alpha^{E_i}]\rho(\gamma_i(t_i))[\alpha^{E_i}/X] :: \rho(G'_1)[\alpha/X]) :: \rho(G'_1)[G_i/X]$$

where $E'_i = \text{lift}_{(W'.\Xi_i)}(G_i)$.

Note that $\varepsilon \vdash \Xi; \Delta \vdash \forall X.G \sim \forall X.G'_1$, then $\varepsilon = \langle \forall X.E_1, \forall X.E_2 \rangle$, for some E_1, E_2, K and L . By the Lemma A.109 we know that $\varepsilon_i^\rho \vdash W'.\Xi_i; \Delta \vdash \forall X.\rho(G) \sim \forall X.\rho(G'_1)$, then $\varepsilon_i^\rho = \langle \forall X.E_{i1}, \forall X.E_{i2} \rangle$, where $\forall X.E_{i1} = \rho_i(E_1)$ and $E_{i2} = \rho_i(E_2)$.

Then we have to prove that:

$$(W'', (\varepsilon_1^\rho[\alpha^{E_1}])\rho(\gamma_1(t_1))[\alpha^{E_1}/X] :: \rho(G'_1)[\alpha/X], \\ (\varepsilon_2^\rho[\alpha^{E_2}])\rho(\gamma_2(t_2))[\alpha^{E_2}/X] :: \rho(G'_1)[\alpha/X]) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[G'_1]$$

where $W'' = \downarrow(W' \boxtimes (\alpha, G_1, G_2, R))$.

Note that

$$W''.\Xi_1 \triangleright (\varepsilon_1^\rho \llbracket \alpha^{E_1} \rrbracket)\rho(\gamma_1(t_1)) :: \rho(G'_1)[\alpha/X] \mapsto^{k^*} \\ \Xi_1 \triangleright (\varepsilon_1^\rho \llbracket \alpha^{E_1} \rrbracket)v_{1f} \mapsto^k \\ \Xi_1 \triangleright v_1^*$$

Let $\rho' = \rho[X \mapsto \alpha]$. We instantiate the premise $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : G$ with W'', ρ' and γ , such that $W'' \in \mathcal{S}[\Xi]$, as $\alpha \in \text{dom}(W'.\kappa[\alpha \mapsto R])$ then $(W'', \rho') \in \mathcal{D}[\Delta, X]$. Also note that as X is fresh, then $\forall(v_1^*, v_2^*) \in \text{cod}(\gamma)$, such that $\Xi; \Delta; \Gamma \vdash v_i^* : G^*$, $X \notin FV(G^*)$, then it is easy to see that $(W'', \gamma) \in \mathcal{G}_{\rho[X \mapsto \alpha]}[\Gamma]$. Then we know that:

$$(W'', \rho'(\gamma_1(t_1)), \rho'(\gamma_2(t_2))) \in \mathcal{J}_{\rho'}[G]$$

But note that:

$$\rho'(\gamma_i(t_i)) = \rho[\alpha/X](\gamma_i(t_i)) = \rho(\gamma_i(t_i))[\alpha^{E_i}/X]$$

Then we have that:

$$(W'', \rho(\gamma_1(t_1))[\alpha^{E_1}/X], \rho(\gamma_2(t_2))[\alpha^{E_2}/X]) \in \mathcal{J}_{\rho[\alpha/X]}[G]$$

If the following term reduces to error, then the result follows immediately.

$$W''.\Xi_1 \triangleright \rho(\gamma_1(t_1))[\alpha^{E_1}/X]$$

If the above is not true, then the following terms reduce to values ($v_{if} = \varepsilon_{if}u_{if} :: \rho'(G)$) and $\exists W''' \succeq W''$ such that $(W''', v_{1f}, v_{2f}) \in \mathcal{V}_{\rho[\alpha \mapsto X]}[G]$ and $W'''.j + k^* = W''.j$.

$$W''.\Xi_i \triangleright \rho(\gamma_i(t_i))[\alpha^{E_i}/X] \longrightarrow^* W'''.\Xi_i \triangleright v_{if}$$

We instantiate the Lemma A.101 with the type G'_1 and the evidence $\langle E_1, E_2 \rangle \vdash \Xi; \Delta, X \vdash G \sim G'_1$ (remember that $\varepsilon = \langle \forall X.E_1, \forall X.E_2 \rangle$). Note that $\varepsilon_i^\rho \llbracket \alpha^{E_i} \rrbracket = \rho[X \mapsto \alpha]_{W'''.\Xi_i}(\langle E_1, E_2 \rangle)$,

$\rho[X \mapsto \alpha](G'_1) = \rho(G'_1)[\alpha/X]$, $W''' \in \mathcal{S}[\Xi]$ and $(W''', \rho[X \mapsto \alpha]) \in \mathcal{D}[\Delta, X]$. Then we obtain that:

$$(W''', (\varepsilon_1^\rho \llbracket \alpha^{E_1} \rrbracket) v_{1f} :: \rho(G'_1)[\alpha/X], (\varepsilon_2^\rho \llbracket \alpha^{E_2} \rrbracket) v_{2f} :: \rho(G'_1)[\alpha/X]) \in \mathcal{T}_{\rho'} \llbracket G'_1 \rrbracket$$

and

$$(\downarrow_k W''', v_1^*, v_2^*) \in \mathcal{T}_\rho \llbracket G'_1 \rrbracket$$

where $(\downarrow_k W''').j + k + k^* = W''.j$ and $v_i^* = (\varepsilon_{if} \circ (\varepsilon_1^\rho \llbracket \alpha^{E_1} \rrbracket)) u_{if} :: \rho(G'_1)[\alpha/X]$, and the result follows immediately.

2. If $G' \in \text{TYPE NAME}$ then $\varepsilon = \langle H_3, \alpha^{E_4} \rangle$. Notice that as α^{E_4} cannot have free type variables therefore H_3 neither. Then $\varepsilon = \rho_i(\varepsilon)$. As α is sync, then let us call $G'' = W.\Xi_i(\alpha)$. We have to prove that:

$$(W, \langle H_3, \alpha^{E_4} \rangle u_1 :: \alpha, \langle H_3, \alpha^{E_4} \rangle u_2 :: \alpha) \in \mathcal{V}_\rho \llbracket \alpha \rrbracket$$

which, by definition of $\mathcal{V}_\rho \llbracket \alpha \rrbracket$, is equivalent to prove that:

$$(\downarrow W, \langle H_3, E_4 \rangle u_1 :: G'', \langle E_3, E_4 \rangle u_2 :: G'') \in \mathcal{V}_\rho \llbracket G'' \rrbracket$$

Then we proceed by case analysis on ε :

- (Case $\varepsilon = \langle H_3, \alpha^{\beta^{E_4}} \rangle$). We know that $\langle H_3, \alpha^{\beta^{E_4}} \rangle \vdash \Xi; \Delta \vdash G^* \sim \alpha$, then by Lemma A.114, $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash G^* \sim G''$. As $\beta^{E_4} \sqsubseteq G''$, then G'' can either be ? or β .
If $G'' = ?$, then by definition of $\mathcal{V}_\rho \llbracket ? \rrbracket$, we have to prove that the resulting values belong to $\mathcal{V}_\rho \llbracket \beta \rrbracket$. Also as $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash G^* \sim ?$, by Lemma A.112, $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash G^* \sim \beta$, and then we proceed just like this case once again (this is process is finite as there are no circular references by construction and it ends up in something different to a type name). If $G'' = \beta$ we use an analogous argument as for $G'' = ?$.
- (Case $\varepsilon = \langle H_3, \alpha^{H_4} \rangle$). We have to prove that

$$(\downarrow W, \langle H_3, H_4 \rangle u_1 :: G'', \langle H_3, H_4 \rangle u_2 :: G'') \in \mathcal{V}_\rho \llbracket G'' \rrbracket$$

By Lemma A.114, $\langle H_3, H_4 \rangle \vdash \Xi; \Delta \vdash G^* \sim G''$. Then if $G'' = ?$, we proceed as the case $G' = ?$, with the evidence $\varepsilon = \langle H_3, H_4 \rangle$. If $G'' \in \text{HEAD TYPE}$, we proceed as the previous case where $G' = \forall X.G$, and the evidence $\varepsilon = \langle H_3, H_4 \rangle$.

Also, we have to prove that $(\forall \Xi', \varepsilon', G_1^*$, such that $\varepsilon'.n = k$, $\varepsilon' = \langle \alpha^{E_1^{**}}, E_2^{**} \rangle$) $(\downarrow W \in \mathcal{S}[\Xi'] \wedge \varepsilon' \vdash \Xi' \vdash \alpha \sim G_1^*)$, we get that

$$(\downarrow_1 W, \varepsilon'(\langle H_3, \alpha^{H_4} \rangle u_1 :: \alpha) :: G_1^*, \varepsilon'(\langle H_4, \alpha^{E_2} \rangle u_2 :: \alpha) :: G_1^*) \in \mathcal{T}_\rho \llbracket G_1^* \rrbracket$$

or what is the same $((\langle H_3, \alpha^{H_4} \rangle \circ \varepsilon')$ fails the result follows immediately)

$$(\downarrow_{1+k} W, (\langle H_3, \alpha^{H_4} \rangle \circ \varepsilon') u_1 :: G_1^*, (\langle H_2, \alpha^{H_4} \rangle \circ \varepsilon') u_2 :: G_1^*) \in \mathcal{V}_\rho \llbracket G_1^* \rrbracket$$

By definition of transitivity and Lemma A.115, we know that

$$\langle H_3, \alpha^{H_4} \rangle \circ \langle \alpha^{E_1^{**}}, E_2^{**} \rangle = \langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle$$

We know that $\langle E_1^{**}, E_2^{**} \rangle \vdash \Xi' \vdash G'' \sim G_1^*$. Since $\langle E_1^{**}, E_2^{**} \rangle \vdash \Xi \vdash G'' \sim G_1^*$, $\downarrow_1 W \in \mathcal{S}[\Xi']$, $(\downarrow_1 W, \langle H_3, H_4 \rangle u_1 :: G'', \langle H_1, H_4 \rangle u_2 :: G'') \in \mathcal{V}_\rho \llbracket G'' \rrbracket$, by Lemma A.101, we know

that (since $(\langle H_3, \alpha^{H_4} \rangle \circ \varepsilon')$ does not fail then $(\langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle)$ also does not fail by the transitivity rules)

$$(\downarrow_{1+k}W, (\langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle)u_1 :: G_1^*, (\langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle)u_2 :: G_1^*) \in \mathcal{V}_\rho[[G_1^*]]$$

The result follows immediately.

3. If $G' = ?$ we have the following cases:

- $(G' = ?, \varepsilon = \langle H_3, H_4 \rangle)$. By the definition of $\mathcal{V}_\rho[[?]]$ in this case we have to prove that:

$$(W, \rho_1(\varepsilon)u_1 :: \rho(G), \rho_2(\varepsilon)u_2 :: \rho(G)) \in \mathcal{V}_\rho[[\text{const}(H_4)]]$$

but as $\text{const}(H_4) = \forall X.?$, we proceed just like the case where $G' = \forall X.G'_1$, where $G'_1 = ?$.

- $(G' = ?, \varepsilon = \langle H_3, \alpha^{E_4} \rangle)$. Notice that as α^{E_4} cannot have free type variables therefore E_3 neither. Then $\varepsilon = \rho_i(\varepsilon)$. By the definition of $\mathcal{V}_\rho[[?]]$ we have to prove that

$$(W, \langle H_3, \alpha^{E_4} \rangle u_1 :: \alpha, \langle H_3, \alpha^{E_4} \rangle u_2 :: \alpha) \in \mathcal{V}_\rho[[\alpha]]$$

Note that by Lemma A.112 we know that $\varepsilon \vdash \Xi; \Delta \vdash G^* \sim \alpha$. Then we proceed just like the case $G' \in \text{TYPE_NAME}$.

□

Proposition A.87 (Compatibility-EpairU) *If $\Xi; \Delta; \Gamma \vdash \pi_1(\varepsilon)u_1 :: G_1 \preceq \pi_1(\varepsilon)u'_1 :: G_1 : G_1$, $\Xi; \Delta; \Gamma \vdash \pi_2(\varepsilon)u'_2 :: G_2 \preceq \pi_2(\varepsilon)u_2 :: G_2 : G_2$, and $\varepsilon \Vdash \Xi; \Delta \vdash G_1 \times G_2 \sim G$ then:*

$$\Xi; \Delta; \Gamma \vdash \varepsilon \langle u_1, u_2 \rangle :: G \preceq \varepsilon \langle u'_1, u'_2 \rangle :: G : G$$

PROOF. Straightforward as the definition of related pairs depends on a weaker property of the premise: $\Xi; \Delta; \Gamma \vdash \pi_1(\varepsilon)u_1 :: G_1 \preceq \pi_1(\varepsilon)u'_1 :: G_1 : G_1$ and $\Xi; \Delta; \Gamma \vdash \pi_2(\varepsilon)u'_2 :: G_2 \preceq \pi_2(\varepsilon)u_2 :: G_2 : G_2$. □

Proposition A.88 (Compatibility-Epair) *If $\Xi; \Delta; \Gamma \vdash t_1 \preceq t'_1 : G_1$ and $\Xi; \Delta; \Gamma \vdash t_2 \preceq t'_2 : G_2$, then $\Xi; \Delta; \Gamma \vdash \langle t_1, t_2 \rangle \preceq \langle t'_1, t'_2 \rangle : G_1 \times G_2$.*

PROOF. We proceed by induction on subterms t_i , analogous to the function application case, but using Prop A.87 instead. □

Proposition A.89 (Compatibility-Ex) *If $x : G \in \Gamma$ and $\Xi; \Delta \vdash \Gamma$ then $\Xi; \Delta; \Gamma \vdash x \preceq x : G$.*

PROOF. First, we are required to show $\Xi; \Delta; \Gamma \vdash x : G$, which is immediate. Consider arbitrary W, ρ, γ such that $W \in \mathcal{S}[[\Xi]]$, $(W, \rho) \in \mathcal{D}[[\Delta]]$ and $(W, \gamma) \in \mathcal{G}_\rho[[\Gamma]]$. We are required to show that:

$$(W, \rho(\gamma_1(x)), \rho(\gamma_2(x))) \in \mathcal{F}_\rho[[G]]$$

Consider arbitrary i, v_1 and Ξ_1 such that $i < W.j$ and $W.\Xi_1 \triangleright \rho(\gamma_1(x)) \longrightarrow^i \Xi_1 \triangleright v_1$. Since $\rho(\gamma_1(x)) = \gamma_1(x)$ and $\gamma_1(x)$ is already a value, we have $i = 0$ and $\gamma_1(x) = v_1$. We are required to show that exists Ξ_2, v_2 such that $W.\Xi_2 \triangleright \gamma_2(x) \longrightarrow^* \Xi_2 \triangleright v_2$ which is immediate (since $\rho(\gamma_2(x)) = \gamma_2(x)$ is a value and $\Xi_2 = W.\Xi_2$). Also, we are required to show that $\exists W'$, such that $W'.j + i = W.j \wedge W' \succeq W \wedge W'.\Xi_1 = \Xi_1 \wedge W'.\Xi_2 = \Xi_2 \wedge (W', \gamma_1(x), \gamma_2(x)) \in \mathcal{V}_\rho[[G]]$. Let $W' = W$, then $(W, \gamma_1(x), \gamma_2(x)) \in \mathcal{V}_\rho[[G]]$ because of the definition of $(W, \gamma) \in \mathcal{G}_\rho[[\Gamma]]$. \square

Proposition A.90 (Compatibility-Easc) *If $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : G$ and $\varepsilon \vdash \Xi; \Delta \vdash G \sim G'$ then $\Xi; \Delta; \Gamma \vdash \varepsilon t_1 :: G' \preceq \varepsilon t_2 :: G' : G'$.*

PROOF. First we are required to prove that $\Xi; \Delta; \Gamma \vdash \varepsilon t_i :: G' : G'$, but by $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : G$ we already know that $\Xi; \Delta; \Gamma \vdash t_i : G$, therefore:

$$\text{(Easc)} \frac{\Xi; \Delta; \Gamma \vdash t_i : G \quad \varepsilon \vdash \Xi; \Delta \vdash G \sim G'}{\Xi; \Delta; \Gamma \vdash \varepsilon t_i :: G' : G'}$$

Consider arbitrary W, ρ, γ such that $W \in \mathcal{S}[[\Xi]]$, $(W, \rho) \in \mathcal{D}[[\Delta]]$ and $(W, \gamma) \in \mathcal{G}_\rho[[\Gamma]]$. We are required to show that:

$$(W, \rho(\gamma_1(\varepsilon t_1 :: G')), \rho(\gamma_2(\varepsilon t_2 :: G'))) \in \mathcal{T}_\rho[[G']]$$

Let's suppose that $\varepsilon_1^p.n = k$. By definition of substitutions $\rho(\gamma_i(\varepsilon t_i :: G')) = \rho(\varepsilon)\rho(\gamma_i(t_i)) :: \rho(G')$, therefore we have to prove that:

$$(W, \rho(\varepsilon)\rho(\gamma_1(t_1)) :: \rho(G'), \rho(\varepsilon)\rho(\gamma_2(t_2)) :: \rho(G')) \in \mathcal{T}_\rho[[G']]$$

First we have to prove that:

$$W.\Xi_i \vdash \rho(\varepsilon)\rho(\gamma_i(t_i)) :: \rho(G') : G'$$

As we know that $\Xi; \Delta; \Gamma \vdash \varepsilon t_i :: G' : G'$, by Lemma A.110 the result follows immediately.

Second, consider arbitrary $i < W.j, \Xi_1$. Either there exist v_1 such that:

$$W.\Xi_1 \triangleright \rho(\varepsilon)\rho(\gamma_1(t_1)) :: \rho(G') \longmapsto^i \Xi_1 \triangleright v_1$$

or

$$W.\Xi_1 \triangleright \rho(\varepsilon)\rho(\gamma_1(t_1)) :: \rho(G') \longmapsto^i \mathbf{error}$$

Let us suppose that $W.\Xi_1 \triangleright \rho(\gamma_1(t_1)) :: \rho(G') \longmapsto^i \Xi_1 \triangleright v_1$. Hence, by inspection of the operational semantics, it follows that there exist $i_1 + 1 < i$, Ξ_{11} and v_{11} such that:

$$W.\Xi_1 \triangleright \rho(\varepsilon)\rho(\gamma_1(t_1)) :: \rho(G') \longmapsto^{i_1} \Xi_{11} \triangleright \rho(\varepsilon)v_{11} :: \rho(G') \longmapsto^k \Xi_{11} \triangleright v_1$$

We instantiate the hypothesis $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : G$ with W, ρ and γ to obtain that:

$$(W, \rho(\gamma_1(t_1)), \rho(\gamma_2(t_2))) \in \mathcal{T}_\rho[[G]]$$

We instantiate $\mathcal{J}_\rho[[G]]$ with i_1 , Ξ_{11} and v_{11} (note that $i_1 < i < W.j$), hence there exists v_{12} and W_1 , such that $W_1 \succeq W$, $W_1.j + i_1 = W.j$, $W.\Xi_2 \triangleright \rho(\gamma_2(t_2)) \mapsto^* W_1.\Xi_2 \triangleright v_{12}$, $W_1.\Xi_1 = \Xi_{11}$, v_{12} and $(W_1, v_{11}, v_{12}) \in \mathcal{V}_\rho[[G]]$.

Since we have that $(W_1, v_{11}, v_{12}) \in \mathcal{V}_\rho[[G]]$, then it is true that $(W_1, \rho(\varepsilon)v_{11} :: G', \rho(\varepsilon)v_{12} :: G') \in \mathcal{J}_\rho[[G']]$ by the Lemma A.101.

By the inspection of the operational semantics:

$$W.\Xi_1 \triangleright \rho(\varepsilon)\rho(\gamma_1(t_1)) :: \rho(G') \mapsto^{i_1} W_1.\Xi_1 \triangleright \rho(\varepsilon)v_{11} :: \rho(G') \mapsto^k \Xi_1 \triangleright v_1$$

We instantiate $(W_1, \rho(\varepsilon)v_{11} :: G', \rho(\varepsilon)v_{12} :: G') \in \mathcal{J}_\rho[[G']]$ with k , v_1 and Ξ_1 . Therefore there must exist v_2 and W' such that $W' \succeq W_1$ (note that $W' \succeq W$), $W'.j + i_1 + k = W'.j + i = W.j$.

$$W_1.\Xi_2 \triangleright \rho(\varepsilon)v_{12} :: \rho(G') \mapsto^* \Xi_2 \triangleright v_2$$

and $(W', v_1, v_2) \in \mathcal{V}_\rho[[G']]$ then the result follows. \square

Proposition A.91 (Compatibility-Eop) *If $\Xi; \Delta; \Gamma \vdash \bar{t} \preceq \bar{t}' : \bar{G}$ and $ty(op) = \bar{G} \rightarrow G$ then $\Xi; \Delta; \Gamma \vdash op(\bar{t}) \preceq op(\bar{t}') : G$.*

PROOF. Similar to the term application. \square

Proposition A.92 (Compatibility-Eapp) *If $\Xi; \Delta; \Gamma \vdash t_1 \preceq t'_1 : G_{11} \rightarrow G_{12}$ and $\Xi; \Delta; \Gamma \vdash t_2 \preceq t'_2 : G_{11}$ then $\Xi; \Delta; \Gamma \vdash t_1 t_2 \preceq t'_1 t'_2 : G_{12}$.*

PROOF. First, we are required to show that:

$$\Xi; \Delta; \Gamma \vdash t_1 t_2 : G_{12}$$

which follows directly from (Eapp) as $\Xi; \Delta; \Gamma \vdash t_1 : G_{11}$, and $\Xi; \Delta; \Gamma \vdash t_2 : G_2$. Also, we are required to prove that:

$$\Xi; \Delta; \Gamma \vdash t'_1 t'_2 : G_{12}$$

which follows analogously.

Second, consider arbitrary W, ρ, γ such that $W \in \mathcal{S}[[\Xi]]$, $(W, \rho) \in \mathcal{D}[[\Delta]]$ and $(W, \gamma) \in \mathcal{G}_\rho[[\Gamma]]$. We are required to show that:

$$(W, \rho(\gamma_1(t_1 t_2)), \rho(\gamma_2(t'_1 t'_2))) \in \mathcal{J}_\rho[[G_{12}]]$$

Consider arbitrary i , v_1 and Ξ_1 such that $i < W.j$ and:

$$W.\Xi_1 \triangleright \rho(\gamma_1(t_1 t_2)) \longrightarrow^i \Xi_1 \triangleright v_1 \vee W.\Xi_1 \triangleright \rho(\gamma_1(t_1 t_2)) \longrightarrow^i \mathbf{error}$$

Hence, by inspection of the operational semantics, it follows that there exist $i_1 < i$, Ξ_{11} and v_{11} such that:

$$W.\Xi_1 \triangleright \rho(\gamma_1(t_1)) \longrightarrow^{i_1} \Xi_{11} \triangleright v_{11} \vee W.\Xi_1 \triangleright \rho(\gamma_1(t_1)) \longrightarrow^{i_1} \mathbf{error}$$

If $W.\Xi_1 \triangleright \rho(\gamma_1(t_1)) \longrightarrow^{i_1} \mathbf{error}$ then $W.\Xi_1 \triangleright \rho(\gamma_2(t'_1)) \longrightarrow^* \mathbf{error}$ and the result holds immediately. Let us assume that the reduction does not fail. We instantiate the hypothesis $\Xi; \Delta; \Gamma \vdash t_1 \preceq t'_1 : G_{11} \rightarrow G_{12}$ with W , ρ and γ we obtain that:

$$(W, \rho(\gamma_1(t_1))), \rho(\gamma_2(t'_1))) \in \mathcal{F}_\rho[[G_{11} \rightarrow G_{12}]]$$

We instantiate this with i_1 , Ξ_{11} and v_{11} (note that $i_1 < i < W.j$), hence there exists v'_{11} and W_1 , such that $W_1 \succeq W$, $W_1.j + i_1 = W.j$, or what is the same $W_1.j + i_1 = W.j$, $W.\Xi_2 \triangleright \rho(\gamma_2(t'_1)) \longrightarrow^* W_1.\Xi_2 \triangleright v'_{11}$, $W_1.\Xi_1 = \Xi_{11}$ and $(W_1, v_{11}, v'_{11}) \in \mathcal{V}_\rho[[G_{11} \rightarrow G_{12}]]$.

Note that:

$$W.\Xi_1 \triangleright \rho(\gamma_1(t_1 \ t_2)) \longrightarrow^{i_1} \Xi_{11} \triangleright v_{11}(\rho(\gamma_1(t_2))) \longrightarrow^{i-i_1} \Xi_1 \triangleright v_1$$

or

$$W.\Xi_1 \triangleright \rho(\gamma_1(t_1 \ t_2)) \longrightarrow^{i_1} \Xi_{11} \triangleright v_{11}(\rho(\gamma_1(t_2))) \longrightarrow^{i-i_1} \mathbf{error}$$

Hence, by inspection of the operational semantics, it follows that there exist $i_2 < i - i_1$, Ξ_{22} and v_{22} such that:

$$\Xi_{11} \triangleright \rho(\gamma_1(t_2)) \longrightarrow^{i_2} \Xi_{22} \triangleright v_{22} \vee \Xi_{11} \triangleright \rho(\gamma_1(t_2)) \longrightarrow^{i_2} \mathbf{error}$$

We instantiate the hypothesis $\Xi; \Delta; \Gamma \vdash t_2 \preceq t'_2 : G_{11}$ with (W_1) , ρ and γ , then we obtain that:

$$(W_1, \rho(\gamma_1(t_2)), \rho(\gamma_2(t'_2))) \in \mathcal{F}_\rho[[G_{11}]]$$

If $\Xi_{11} \triangleright \rho(\gamma_1(t_2)) \longrightarrow^{i_2} \mathbf{error}$ then we instantiate with Ξ_{22} and $\Xi_{22} \triangleright \rho(\gamma_2(t'_2)) \longrightarrow^* \mathbf{error}$ and the result holds immediately. Let us assume that the reduction does not fail. We instantiate this with i_2 (note that $i_2 < i - i_1 < W_1.j = W.j - i_1$), Ξ_{22} and v_{22} , hence there exists v'_{22} and W_2 , such that $W_2.\Xi_1 = \Xi_{22}$, $W_2 \succeq W_1$, or what is the same, $W_2 \succeq W_1$, $W_2.j = W_1.j - i_2$ ($W_2.j + i_2 + i_1 = W.j$) and

$$W_1.\Xi_2 \triangleright \rho(\gamma_2(t'_2)) \longrightarrow^* W_2.\Xi_2 \triangleright v'_{22}$$

and $(W_2, v_{22}, v'_{22}) \in \mathcal{V}_\rho[[G_{11}]]$.

Note that:

$$W.\Xi_1 \triangleright \rho(\gamma_1(t_1 \ t_2)) \longrightarrow^{i_1} \Xi_{11} \triangleright v_{11}(\rho(\gamma_1(t_2))) \longrightarrow^{i_2} \Xi_{22} \triangleright v_{11} \ v_{22} \longrightarrow^{i-i_1-i_2} \Xi_1 \triangleright v_1$$

Since $(W_1, v_{11}, v'_{11}) \in \mathcal{V}_\rho[[G_{11} \rightarrow G_{12}]]$, we instantiate this with W_2 , $\rho(G_{11} \rightarrow G_{12})$, v_{22} and v'_{22} (note that $(W_2, v_{22}, v'_{22}) \in \mathcal{V}_\rho[[G_{11}]]$, $(\downarrow_1 W_2, v_{22}, v'_{22}) \in \mathcal{V}_\rho[[G_{11}]]$ and $W_2 \succeq W_1$). Then $(W_2, v_{11} \ v_{22}, v'_{11} \ v'_{22}) \in \mathcal{F}_\rho[[G_2]]$.

Since $(W_2, v_{11} \ v_{22}, v'_{11} \ v'_{22}) \in \mathcal{F}_\rho[[G_2]]$, we instantiate this with $i - i_1 - i_2$ (note that $i - i_1 - i_2 < W_2.j = W.j - i_1 - i_2$ since $i < W.j$), v_1 and Ξ_1 .

If $W_2.\Xi_1 \triangleright v_{11} \ v_{22} \xrightarrow{i-i_1-i_2} \mathbf{error}$ then $W_2.\Xi_2 \triangleright v'_{11} \ v'_{22} \xrightarrow{*} \mathbf{error}$ and the result holds. Let us assume that the reduction does not fail. Hence there exists v_2 and W' , such that $W' \succeq W_2$ (note that $W' \succeq W$), $W'.j = W_2.j - (i - i_1 - i_2) = W.j - i$, $W_2.\Xi_2 \triangleright v'_{11} \ v'_{22} \xrightarrow{*} W'.\Xi_2 \triangleright v_2$, $W'.\Xi_1 = \Xi_1$ and $(W', v_1, v_2) \in \mathcal{V}_\rho[[G_{12}]]$, then the proof is complete. \square

Lemma A.93 (Compatibility-EappG) *If $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : \forall X.G$ and $\Xi; \Delta \vdash G'$, then $\Xi; \Delta; \Gamma \vdash t_1 [G'] \preceq t_2 [G'] : G[G'/X]$.*

PROOF. First we are required to prove that $\Xi; \Delta; \Gamma \vdash t_i [G'] : G[G'/X]$, but by $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : \forall X.G$ we already know that $\Xi; \Delta; \Gamma \vdash t_i : \forall X.G$, therefore:

$$\text{(EappG)} \frac{\Xi; \Delta; \Gamma \vdash t_i : \forall X.G \quad \Xi; \Delta \vdash G'}{\Xi; \Delta; \Gamma \vdash t_i [G'] : G[G'/X]}$$

Consider arbitrary W, ρ, γ such that $W \in \mathcal{S}[[\Xi]]$, $(W, \rho) \in \mathcal{D}[[\Delta]]$ and $(W, \gamma) \in \mathcal{G}_\rho[[\Gamma]]$. We are required to show that:

$$(W, \rho(\gamma_1(t_1[G'])), \rho(\gamma_2(t_2[G']))) \in \mathcal{F}_\rho[[G[G'/X]]]$$

But by definition of substitutions $\rho(\gamma_i(t_i[G'])) = \rho(\gamma_i(t_i))[\rho(G')]$, therefore we have to prove that:

$$(W, \rho(\gamma_1(t_1))[\rho(G')], \rho(\gamma_2(t_2))[\rho(G')]) \in \mathcal{F}_\rho[[G[G'/X]]]$$

First we have to prove that:

$$W.\Xi_i \vdash \rho(\gamma_i(t_i))[\rho(G')] : \rho(G)[\rho(G')/X]$$

As we know that $\Xi; \Delta; \Gamma \vdash t_i [G'] : G[G'/X]$, by Lemma A.110 the result follows immediately. Second, consider arbitrary $i < W.j$ and Ξ_1 . Either there exist v_1 such that $W.\Xi_1 \triangleright \rho(\gamma_1(t_1))[\rho(G')] \mapsto^i \Xi_1 \triangleright v_1$ or $W.\Xi_1 \triangleright \rho(\gamma_1(t_1))[\rho(G')] \mapsto^i \Xi_1 \triangleright \mathbf{error}$. First, let us suppose that:

$$W.\Xi_1 \triangleright \rho(\gamma_1(t_1))[\rho(G')] \mapsto^i \Xi_1 \triangleright v_1$$

Hence, by inspection of the operational semantics, it follows that there exist $i_1 < i$, and v_{11} such that

$$W.\Xi_1 \triangleright \rho(\gamma_1(t_1))[\rho(G')] \mapsto^{i_1} \Xi_{11} \triangleright v_{11}[\rho(G')]$$

We instantiate the premise $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : \forall X.G$ with W, ρ and γ to obtain that:

$$(W, \rho(\gamma_1(t_1)), \rho(\gamma_2(t_2))) \in \mathcal{F}_\rho[[\forall X.G]]$$

We instantiate $\mathcal{F}_\rho[[\forall X.G]]$ with i_1, Ξ_{11} and v_{11} (note that $i_1 < i < W.j$), hence there exists v_{12} and W_1 , such that $W_1 \succeq W$, $W_1.j = W.j - i_1$, $W.\Xi_2 \triangleright \rho(\gamma_2(t_2)) \mapsto^* W_1.\Xi_2 \triangleright v_{12}$, $W_1.\Xi_1 = \Xi_{11}$, v_{12} and:

$$(W_1, v_{11}, v_{12}) \in \mathcal{V}_\rho[[\forall X.G]]$$

Then by inspection of the operational semantics:

$$\begin{aligned} W.\Xi_i \triangleright \rho(\gamma_i(t_i))[\rho(G')] &\mapsto^* W_1.\Xi_i \triangleright v_{1i}[\rho(G')] \\ &\mapsto W_1.\Xi_i, \alpha := \rho(G') \triangleright \varepsilon_i(\varepsilon'_i t'_i :: \rho(G)[\alpha/X]) :: \rho(G)[\rho(G')/X] \end{aligned}$$

for some $\varepsilon_1, \varepsilon_2, \varepsilon'_1, \varepsilon'_2, t'_i$ and $\alpha \notin \text{dom}(W_1.\Xi_i)$. Let us call $t''_i = (\varepsilon'_i t'_i :: \rho(G)[\alpha/X])$. We instantiate $\mathcal{V}_\rho[\llbracket \forall X.G \rrbracket]$ with $\alpha, t''_i, \rho(G'), R = \mathcal{V}_\rho[\llbracket G' \rrbracket], \varepsilon_1, \varepsilon_2$ and W_1 .

Then $(W'_1, t''_1, t''_2) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[\llbracket G \rrbracket]$, where $W'_1 = (\downarrow W_1) \boxtimes (\alpha, \rho(G'), \rho(G'), \mathcal{V}_\rho[\llbracket G' \rrbracket])$.

We instantiate $\mathcal{J}_{\rho[X \mapsto \alpha]}[\llbracket G \rrbracket]$ with i_2, Ξ_1, v'_1 , such that

$$W_1.\Xi_1 \triangleright (\varepsilon'_1 t'_1 :: \rho(G)[\alpha/X]) \mapsto^{i_2} \Xi_1 \triangleright v'_1$$

Note that $i_2 < W'_1.j = W.j - i_1 - 1$, since $i < W.j$. Therefore there must exist v'_2 , and W' such that

$$W' \succeq W'_1 \text{ (note that } W' \succeq W), W'.j + i_1 + 1 + i_2 = W.j - i,$$

$$W_1.\Xi_2 \triangleright (\varepsilon'_2 t'_2 :: \rho(G)[\alpha/X]) \mapsto^* W'.\Xi_2 \triangleright (\varepsilon'_2 v''_2 :: \rho(G)[\alpha/X]) \mapsto W'.\Xi_2 \triangleright v'_2$$

$$W'.\Xi_1 = \Xi_1 \text{ and } (W', v'_1, v'_2) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[\llbracket G \rrbracket].$$

Notice that t_i reduce to a type abstraction of the form $v_{1i} = \langle \forall X.E_{i1}, \forall X.E_{i2} \rangle \wedge X.t''_i :: \forall X.\rho(G)$. Let us call $v'_i = \varepsilon''_i u''_i :: \rho(G)[\alpha/X]$, as $\pi_2(\varepsilon''_i) \equiv \pi_2(\varepsilon'_i)$, then $G_p = \text{unlift}(\pi_2(\varepsilon''_i))$, then $E_i = \text{lift}_{W_2.\Xi_i}(G_p)$, and $E'_i = \text{lift}_{W_1.\Xi_i}(\rho(G'))$, and $\varepsilon_i = \langle E_i[\alpha^{E'_i}/X], E_i[E'_i/X] \rangle$. Then as $(W', v'_1, v'_2) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[\llbracket G \rrbracket]$ by Lemma A.98,

$$(\downarrow_k W', (\varepsilon''_1 \circledast \varepsilon_1) u''_1 :: \rho(G)[\rho(G')/X], (\varepsilon''_2 \circledast \varepsilon_2) u''_2 :: \rho(G)[\rho(G')/X]) \in \mathcal{V}_\rho[\llbracket G[G'/X] \rrbracket]$$

where $\varepsilon_1.n = k$. Let us call $v_i = (\varepsilon''_i \circledast \varepsilon_i) u''_i :: \rho(G)[\rho(G')/X]$. Where the lemma holds by instantiating $\mathcal{J}_\rho[\llbracket G[G'/X] \rrbracket]$ with $\Xi_1, v_1, i = k$ and therefore $W'.\Xi_1 \triangleright \varepsilon_1 v'_1 :: \rho(G)[\rho(G')/X] \mapsto^k W'.\Xi_1 \triangleright v_1$. Then there must exists some v_2 such that $W'.\Xi_2 \triangleright \varepsilon_2 v'_2 :: \rho(G)[\rho(G')/X] \mapsto W'.\Xi_2 \triangleright v_2$, and the result follows. □

Proposition A.94 (Compatibility-Epair1) *If $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : G_1 \times G_2$ then $\Xi; \Delta; \Gamma \vdash \pi_1(t_1) \preceq \pi_1(t_2) : G_1$.*

PROOF. Similar to the function application case, using the definition of related pairs instead. □

Proposition A.95 (Compatibility-Epair2) *If $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : G_1 \times G_2$ then $\Xi; \Delta; \Gamma \vdash \pi_2(t_1) \preceq \pi_2(t_2) : G_2$.*

PROOF. Similar to the function application case, using the definition of related pairs instead. □

Lemma A.96 *Let $E_i = \text{lift}_{\Xi_i}(G_p)$ for some $G_p \sqsubseteq G$, $\langle E_{i1}, E_{i2} \rangle \Vdash \Xi_i \vdash G_u \sim G$, and $E_{12} \equiv E_{22}$, then*

$$\langle E_{11}, E_{12} \rangle \circledast \langle E_1, E_1 \rangle \iff \langle E_{21}, E_{22} \rangle \circledast \langle E_2, E_2 \rangle.$$

PROOF. Note that by definition $E_1 \equiv E_2$. Also, $\forall \alpha^E \in FTN(E_i), E = \text{lift}_{\Xi_i}(\Xi_i(\alpha))$. Then we prove the \Rightarrow direction (the other is analogous), by induction on the structure of the evidences $\langle E_{i1}, E_{i2} \rangle$. We skip cases where $E_i = ?$ or $E_{i1} = ?$, as the result is trivial (combination never fails).

Case ($\langle E_{11}, E_{12} \rangle = \langle E_{11}, \alpha^{E'_{12}} \rangle$). Then $\langle E_{21}, E_{22} \rangle = \langle E_{21}, \alpha^{E'_{22}} \rangle$, and $E_i = \langle \alpha^{E'_i}, \alpha^{E'_i} \rangle$, where $E'_i = \text{lift}_{\Xi_i}(\Xi_i(\alpha))$, and therefore $E'_{i2} \sqsubseteq E'_i$. And then by Lemma A.115, the result holds immediately as both combinations are defined.

Case ($\langle E_{11}, E_{12} \rangle = \langle E_{11}, B \rangle$). Then $\langle E_{21}, E_{22} \rangle = \langle E_{12}, B \rangle$, and $\langle E_i, E_i \rangle = \langle B, B \rangle$, and the result trivially holds.

Case ($\langle E_{11}, E_{12} \rangle = \langle \alpha^{E'_{11}}, E_{12} \rangle$). The result holds by de inspection of consistent transitivity rule (sealR) and induction on evidence $\langle E'_{i1}, E_{i2} \rangle$.

Case ($\langle E_{11}, E_{12} \rangle = \langle E_{111} \rightarrow E_{112}, E_{121} \rightarrow E_{122} \rangle$). Then $\langle E_{11}, E_{12} \rangle = \langle E_{111} \rightarrow E_{112}, E_{121} \rightarrow E_{122} \rangle$, and $\langle E_i, E_i \rangle = \langle E'_{i1} \rightarrow E'_{i2}, E'_{i1} \rightarrow E'_{i2} \rangle$. As consistent transitivity is a symmetric relation, then the result holds by induction hypothesis on combinations of evidence $\langle E_{i11} \rightarrow E_{i12} \rangle \circ \langle E'_{i1}, E'_{i1} \rangle$ and $\langle E_{i21} \rightarrow E_{i22} \rangle \circ \langle E'_{i2}, E'_{i2} \rangle$.

For the other cases we proceed analogous to the function case. □

Proposition A.97 *If* $(W, v_1, v_2) \in \mathcal{V}_\rho \llbracket G \rrbracket$ *and* $W' \succeq W$ *then* $(W, v_1, v_2) \in \mathcal{V}_\rho \llbracket G \rrbracket$.

Proposition A.98 (Compositionality) *If*

- $W.\Xi_i(\alpha) = \rho(G')$ *and* $W.\kappa(\alpha) = \mathcal{V}_\rho \llbracket G' \rrbracket$,
- $E'_i = \text{lift}_{W.\Xi_i}(\rho(G'))$,
- $E_i = \text{lift}_{W.\Xi_i}(G_p)$ *for some* $G_p \sqsubseteq \rho(G)$,
- $\rho' = \rho[X \mapsto \alpha]$,
- $\varepsilon_i = \langle E_i[\alpha^{E'_i}/X], E_i[E'_i/X] \rangle$, *such that* $\varepsilon_i \vdash W.\Xi_i \vdash \rho(G[\alpha/X]) \sim \rho(G[G'/X])$, *and*
- $\varepsilon_i^{-1} = \langle E_i[E'_i/X], E_i[\alpha^{E'_i}/X] \rangle$, *such that* $\varepsilon_i^{-1} \vdash W.\Xi_i \vdash \rho(G[G'/X]) \sim \rho(G[\alpha/X])$, *then*

1.

$$(W, \varepsilon'_1 u_1 :: \rho'(G), \varepsilon'_2 u_2 :: \rho'(G)) \in \mathcal{V}_{\rho'} \llbracket G \rrbracket \Rightarrow \\ (W, \varepsilon_1(\varepsilon'_1 u_1 :: \rho(G)) :: \rho(G[G'/X]), \varepsilon_2(\varepsilon'_2 u_2 :: \rho(G)) :: \rho(G[G'/X])) \in \mathcal{F}_\rho \llbracket G[G'/X] \rrbracket$$

2.

$$(W, \varepsilon'_1 u_1 :: \rho(G[G'/X]), \varepsilon'_2 u_2 :: \rho(G[G'/X])) \in \mathcal{V}_\rho \llbracket G[G'/X] \rrbracket \Rightarrow \\ (W, \varepsilon_1^{-1}(\varepsilon'_1 u_1 :: \rho(G[G'/X])) :: \rho'(G), \varepsilon_2^{-1}(\varepsilon'_2 u_2 :: \rho(G[G'/X])) :: \rho'(G)) \in \mathcal{F}_{\rho'} \llbracket G \rrbracket$$

PROOF. We proceed by induction on G . Let suppose that $\varepsilon_1.n = k$, $\varepsilon_1^{-1}.n = l$ and $\varepsilon'_1.n = m$. Let $v_i = \varepsilon'_i u_i :: \rho'(G)$. We prove (1) first.

Case (Type Variable X: $G = X$). Let $v_i = \langle H_{i1}, \alpha^{E_{i2}} \rangle u_i :: \alpha$. Then we know that

$$(W, \langle H_{11}, \alpha^{E_{12}} \rangle u_1 :: \alpha, \langle H_{21}, \alpha^{E_{22}} \rangle u_2 :: \alpha) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[\![X]\!]$$

which is equivalent to

$$(W, \langle H_{11}, \alpha^{E_{12}} \rangle u_1 :: \alpha, \langle H_{21}, \alpha^{E_{22}} \rangle u_2 :: \alpha) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[\![\alpha]\!]$$

As $W.\Xi_i(\alpha) = \rho(G')$ and $W.\kappa(\alpha) = \mathcal{V}_{\rho}[\![G']\!]$, we know that:

$$(\downarrow_1 W, \langle H_{11}, E_{12} \rangle u_1 :: \rho(G'), \langle H_{21}, E_{22} \rangle u_2 :: \rho(G')) \in \mathcal{V}_{\rho}[\![G']\!]$$

Then $\varepsilon_i \vdash W.\Xi_i \vdash \alpha \sim \rho(G')$, and ε_i has to have the form $\varepsilon_i = \langle \alpha^{E'_i}, E'_i \rangle$. As $E'_i = \text{lift}_{W.\Xi_i}(\rho(G'))$ (initial evidence for α), then $E_{i2} \sqsubseteq E'_i$, and therefore by Lemma A.115: $\langle H_{i1}, \alpha^{E_{i2}} \rangle \mathbin{\text{;}} \langle \alpha^{E'_i}, E'_i \rangle = \langle H_{i1}, E_{i2} \rangle$, and then we have to prove that

$$(\downarrow_k W, \langle H_{11}, E_{12} \rangle u_1 :: \rho(G'), \langle H_{21}, E_{22} \rangle u_2 :: \rho(G')) \in \mathcal{V}_{\rho}[\![G']\!]$$

which follow by Lemma A.97 and the fact that $k > 0$.

Case (Type Variable Y: $G = Y$). Let $v_i = \langle H_{i1}, \beta^{E_{i2}} \rangle u_i :: \beta$, where $\rho'(Y) = \beta$. Then we know that

$$(W, \langle H_{11}, \beta^{E_{12}} \rangle u_1 :: \beta, \langle H_{21}, \beta^{E_{22}} \rangle u_2 :: \beta) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[\![Y]\!]$$

which is equivalent to

$$(W, \langle H_{11}, \beta^{E_{12}} \rangle u_1 :: \beta, \langle H_{21}, \beta^{E_{22}} \rangle u_2 :: \beta) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[\![\beta]\!]$$

Then $\varepsilon_i \vdash W.\Xi_i \vdash \beta \sim \beta$, and ε_i has to have the form $\varepsilon_i = \langle \beta^{E'_i}, \beta^{E'_i} \rangle$, and $\beta^{E'_i} = \text{lift}_{W.\Xi_i}(\beta)$. By Lemma A.96, we assume that both combinations of evidence are defined (otherwise the result holds immediately). Therefore, by Lemma A.115, we know that

$$\langle H_{i1}, \beta^{E_{i2}} \rangle \mathbin{\text{;}} \langle \beta^{E'_i}, \beta^{E'_i} \rangle = \langle H_{i1}, \beta^{E_{i2}} \rangle$$

Then we have to prove that

$$(\downarrow_k W, \langle H_{11}, \beta^{E_{12}} \rangle u_1 :: \beta, \langle H_{21}, \beta^{E_{22}} \rangle u_2 :: \beta) \in \mathcal{V}_{\rho}[\![\beta]\!]$$

which follows Lemma A.97.

Case (Unknown Type: $G = ?$). Let $v_i = \langle H_{i1}, E_{i2} \rangle u_i :: ?$. Then by definition of $\mathcal{V}_{\rho}[\![?]\!]$, let $G'' = \text{const}(E_{i2})$ (where $G'' \neq ?$). Then we know

$$(W, \langle H_{11}, E_{12} \rangle u_1 :: G'', \langle H_{21}, E_{22} \rangle u_2 :: G'') \in \mathcal{V}_{\rho}[\![G'']\!]$$

We are required to prove that:

$$(W, \varepsilon_1(\langle H_{11}, E_{12} \rangle u_1 :: ?) :: ?, \varepsilon_2(\langle H_{21}, E_{22} \rangle u_2 :: ?) :: ?) \in \mathcal{T}_{\rho}[\![?]\!]$$

If $\varepsilon_i = \langle ?, ? \rangle$, then, $\langle H_{i1}, E_{i2} \rangle \mathbin{\text{;}} \langle ?, ? \rangle = \langle H_{i1}, E_{i2} \rangle$, by Lemma A.115, the result holds immediately.

If $\varepsilon_i \neq \langle ?, ? \rangle$. Then we proceed similar to the other cases where $G \neq ?$. Note that we know that

$$(W, \langle H_{11}, E_{12} \rangle u_1 :: G'', \langle H_{21}, E_{22} \rangle u_2 :: G'') \in \mathcal{V}_{\rho}[\![G'']\!]$$

where $G'' \neq ?$ and we are required to prove that

$$(W, \varepsilon_1(\langle H_{11}, E_{12} \rangle u_1 :: G'') :: G'', \varepsilon_2(\langle H_{21}, E_{22} \rangle u_2 :: G'')) \in \mathcal{V}_{\rho}[\![G'']\!]$$

Case (**Function Type:** $G = G_1 \rightarrow G_2$). We know that

$$(W, v_1, v_2) \in \mathcal{V}_{\rho'} \llbracket G_1 \rightarrow G_2 \rrbracket$$

Then we have to prove that

$$\begin{aligned} & (\downarrow_k W, (\varepsilon'_1 \circledast \varepsilon_1)(\lambda x : G'_1.t_1) :: \rho(G_1[G'/X]) \rightarrow \rho(G_2[G'/X]), \\ & (\varepsilon'_2 \circledast \varepsilon_2)(\lambda x : G'_2.t_2) :: \rho(G_1[G'/X]) \rightarrow \rho(G_2[G'/X])) \in \mathcal{V}_{\rho} \llbracket G_1[G'/X] \rightarrow G_2[G'/X] \rrbracket \end{aligned}$$

Let us call $v''_i = (\varepsilon'_i \circledast \varepsilon_i)(\lambda x : G'_i.t_i) :: \rho'(G_1) \rightarrow \rho'(G_2)$. By unfolding, we have to prove that

$$\forall W' \succeq (\downarrow_k W). \forall v'_1, v'_2. (\downarrow_1 W', v'_1, v'_2) \in \mathcal{V}_{\rho} \llbracket G_1[G'/X] \rrbracket \Rightarrow (W', v''_1 v'_1, v''_2 v'_2) \in \mathcal{J}_{\rho} \llbracket G_2[G'/X] \rrbracket$$

Suppose that $v'_i = \varepsilon''_i u'_i :: \rho(G_1[G'/X])$, by inspection of the reduction rules, we know that

$$W'. \Xi_i \triangleright v''_i v'_i \mapsto^* W'. \Xi_i \triangleright (\text{cod}(\varepsilon'_i) \circledast \text{cod}(\varepsilon_i)) t_i [(\varepsilon''_i \circledast (\text{dom}(\varepsilon_i) \circledast \text{dom}(\varepsilon'_i))) u'_i :: G'_i] / x :: \rho(G_2[G'/X])$$

This is equivalent by Lemma A.103,

$$W'. \Xi_i \triangleright v''_i v'_i \mapsto^* W'. \Xi_i \triangleright (\text{cod}(\varepsilon'_i) \circledast \text{cod}(\varepsilon_i)) t_i [(\varepsilon''_i \circledast \text{dom}(\varepsilon_i) \circledast \text{dom}(\varepsilon'_i)) u'_i :: G'_i] / x :: \rho(G_2[G'/X])$$

Therefore, we know that

$$\begin{aligned} & W'. \Xi_1 \triangleright v''_1 v'_1 \mapsto^{m+k+1} \\ & W'. \Xi_1 \triangleright (\text{cod}(\varepsilon'_1) \circledast \text{cod}(\varepsilon_1)) t_1 [(\varepsilon''_1 \circledast (\text{dom}(\varepsilon_1) \circledast \text{dom}(\varepsilon'_1))) u'_1 :: G'_1] / x :: \rho(G_2[G'/X]) \mapsto^{k^*} \\ & \Xi_1 \triangleright (\text{cod}(\varepsilon'_1) \circledast \text{cod}(\varepsilon_1)) v_{1f} :: \rho(G_2[G'/X]) \mapsto^{m+k} \\ & \Xi_1 \triangleright v_1^* \end{aligned}$$

where $v_{1f} = \varepsilon_{1f} u_{1f} :: \rho'(G_2)$ and $v_1^* = \varepsilon_{1f} \circledast (\text{cod}(\varepsilon'_1) \circledast \text{cod}(\varepsilon_1)) u_{1f} :: \rho(G_2[G'/X])$.

Notice that $\text{dom}(\varepsilon_i) \vdash W. \Xi_i \vdash \rho(G_1[G'/X]) \sim \rho(G_1[\alpha/X])$, by Lemma A.96, we assume that both combinations of evidence are defined (otherwise the result holds immediately), then let us assume that $(\varepsilon''_i \circledast \text{dom}(\varepsilon_i))$ is defined. We can use induction hypothesis on v'_i , with evidences $\text{dom}(\varepsilon_i)$. Then we know that $(\downarrow_{k+1} W', (\varepsilon''_1 \circledast \text{dom}(\varepsilon_1)) u'_1 :: \rho'(G_1), (\varepsilon''_2 \circledast \text{dom}(\varepsilon_2)) u'_2 :: \rho'(G_1)) \in \mathcal{V}_{\rho'} \llbracket G_1 \rrbracket$. Let us call $v'''_i = (\varepsilon''_i \circledast \text{dom}(\varepsilon_i)) u'_i :: \rho'(G_1)$.

Now we instantiate

$$(W, v_1, v_2) \in \mathcal{V}_{\rho'} \llbracket G_1 \rightarrow G_2 \rrbracket$$

with $\downarrow_k W'$ and v'''_i and

$$(\downarrow_{k+1} W', (\varepsilon''_1 \circledast \text{dom}(\varepsilon_1)) u'_1 :: \rho'(G_1), (\varepsilon''_2 \circledast \text{dom}(\varepsilon_2)) u'_2 :: \rho'(G_1)) \in \mathcal{V}_{\rho'} \llbracket G_1 \rrbracket$$

to obtain that either both executions reduce to an error (then the result holds immediately), or $\exists W'' \succeq \downarrow_k W'$ such that $W''.j + 2m + 1 + k^* + k = W'.j$ and $(W'', v'_{f1}, v'_{f2}) \in \mathcal{V}_{\rho'} \llbracket G_2 \rrbracket$

$$\begin{aligned} & W'. \Xi_i \triangleright v_i v'''_i \mapsto^* W'. \Xi_i \triangleright \text{cod}(\varepsilon'_i) t_i [(\varepsilon''_i \circledast \text{dom}(\varepsilon_i) \circledast \text{dom}(\varepsilon'_i)) u'_i :: G'_i] / x :: \rho(G_2) \\ & \mapsto^* W''. \Xi_i \triangleright v'_{fi} \end{aligned}$$

Suppose that $v'_{fi} = \varepsilon'_{fi} u_{fi} :: \rho'(G_2)$.

Also, we know that

$$\begin{aligned} W'.\Xi_1 \triangleright v_1 v'_1 &\mapsto^{m+1} \\ W'.\Xi_1 \triangleright \text{cod}(\varepsilon'_1) t_1 [(\varepsilon'_1 \circledast (\text{dom}(\varepsilon_1) \circledast \text{dom}(\varepsilon'_i))) u'_1 :: G'_i/x] :: \rho'(G_2) &\mapsto^{k^*} \\ \Xi_1 \triangleright \text{cod}(\varepsilon'_1) v_{1f} :: \rho'(G_2) &\mapsto^m \\ \Xi_1 \triangleright v'_{1f} & \end{aligned}$$

Then we use induction hypothesis once again using evidences $\text{cod}(\varepsilon_i)$ over v'_{if} (noticing that by Lemma A.96, the combination of evidence either both fail or both are defined), to obtain that,

$$\begin{aligned} (\downarrow_k W'', (\varepsilon_{f1} \circledast \text{cod}(\varepsilon'_1) \circledast \text{cod}(\varepsilon_1)) u_{f1} :: \rho(G_2[G'/X]), \\ (\varepsilon_{f2} \circledast \text{cod}(\varepsilon'_2) \circledast \text{cod}(\varepsilon_2)) u_{f2} :: \rho(G_2[G'/X])) \in \mathcal{V}_\rho[G_2[G'/X]] \end{aligned}$$

and the result holds. Note that $(\downarrow_k W'').j + 1 + 2m + 2k + k^* = W'.j$

Case (Universal Type: $\forall Y.G_1$). We know that

$$(W, v_1, v_2) \in \mathcal{V}_{\rho'}[\forall Y.G_1]$$

Then we have to prove that

$$\begin{aligned} (\downarrow_k W, (\varepsilon'_1 \circledast \varepsilon_1)(\Lambda Y.t_1) :: \forall Y.\rho(G_1[G'/X]), \\ (\varepsilon'_2 \circledast \varepsilon_2)(\Lambda Y.t_2) :: \forall Y.\rho(G_1[G'/X])) \in \mathcal{V}_\rho[\forall Y.G_1[G'/X]] \end{aligned}$$

Let $\varepsilon'_i = \langle \forall Y.E_{i1}, \forall Y.E_{i2} \rangle$ and $\varepsilon_i = \langle \forall Y.E'_{i1}, \forall Y.E'_{i2} \rangle = \langle \forall Y.E''_i[\alpha^{E_i}/X], \forall Y.E''_i[E'_i/X] \rangle$, where $E_i = \forall Y.E''_i$. Let us call $v''_i = (\varepsilon'_i \circledast \varepsilon_i)(\Lambda Y.t_i) :: \forall Y.\rho(G_1[G'/X])$. By unfolding, we have to prove that

$$\begin{aligned} \forall W' \succeq (\downarrow_k W). \forall t''_1, t''_2, G'_1, G'_2, \beta, \varepsilon''_1, \varepsilon''_2. \forall R \in \text{REL}_{W',j}[G'_1, G'_2]. \\ (W'.\Xi_1 \vdash G'_1 \wedge W'.\Xi_2 \vdash G'_2) \wedge \\ W'.\Xi_1 \triangleright v''_1[G'_1] \mapsto W'.\Xi_1, \beta := G'_1 \triangleright \varepsilon''_1 t''_1 :: \rho(G_1)[G'/X][G'_1/Y] \wedge \\ W'.\Xi_2 \triangleright v''_2[G'_2] \mapsto W'.\Xi_2, \beta := G'_2 \triangleright \varepsilon''_2 t''_2 :: \rho(G_1)[G'/X][G'_2/Y] \Rightarrow \\ (W^*, t''_1, t''_2) \in \mathcal{T}_{\rho[Y \mapsto \beta]}[G_1[G'/X]] \end{aligned}$$

where $E_i^* = \text{lift}_{W',\Xi_i}(G'_i)$ and $W^* = \downarrow(W' \boxtimes (\beta, G'_1, G'_2, R))$

By inspection of the reduction rules we know that

$$t''_i = (\langle E_{i1}[\beta^{E_i^*}/Y], E_{i2}[\beta^{E_i^*}/Y] \rangle \circledast \langle E''_i[\alpha^{E_i^*}/X][\beta^{E_i^*}/Y], E''_i[E'_i/X][\beta^{E_i^*}/Y] \rangle) t_i[\beta^{E_i^*}/Y] :: \rho(G_1[G'/X][\beta/Y])$$

Note that $(\langle E_{i1}[\beta^{E_i^*}/Y], E_{i2}[\beta^{E_i^*}/Y] \rangle \circledast \langle E''_i[\alpha^{E_i^*}/X][\beta^{E_i^*}/Y], E''_i[E'_i/X][\beta^{E_i^*}/Y] \rangle).n = m+k$. Therefore, we know that

$$\begin{aligned} W^*. \Xi_1 \triangleright t''_1 &\mapsto^{k^*} \\ \Xi_1 \triangleright (\langle E_{i1}[\beta^{E_i^*}/Y], E_{i2}[\beta^{E_i^*}/Y] \rangle \circledast \langle E''_i[\alpha^{E_i^*}/X][\beta^{E_i^*}/Y], E''_i[E'_i/X][\beta^{E_i^*}/Y] \rangle) & \\ v_{m1} :: \rho(G_1[G'/X][\beta/Y]) &\mapsto^{k+m} \Xi_1 \triangleright v'_1 \end{aligned}$$

By the reduction rule of the type application we know that:

$$W'.\Xi_i \triangleright v_i[G'_i] \mapsto W'.\Xi_i, \beta := G'_i \triangleright \langle E_i^\#[\beta^{E_i^*}/Y], E_i^\#[E'_i/Y] \rangle t'_i :: \rho(G_1[G'/X][G'_i/Y])$$

where $t'_i = (\langle E_{i1}[\beta^{E_i^*}/Y], E_{i2}[\beta^{E_i^*}/Y] \rangle t_i[\beta^{E_i^*}/Y] :: \rho(G_1[G'/X][\beta/Y]))$. Now we instantiate

$$(W, v_1, v_2) \in \mathcal{V}_{\rho'}[\llbracket \forall Y. G_1 \rrbracket]$$

with $W', G'_1, G'_2, R, t'_1, t'_2, \beta$, and evidences $\langle E_{i1}[\beta^{E_i^*}/Y], E_{i2}[E_i^*/Y] \rangle$, to obtain that

$$(W^*, t'_1, t'_2) \in \mathcal{J}_{\rho'[Y \mapsto \beta]}[\llbracket G_1 \rrbracket]$$

then either both executions reduce to an error (then the result holds immediately), or $\exists W'' \succeq W^*, v_{fi}$, such that $(W'', v_{f1}, v_{f2}) \in \mathcal{V}_{\rho'[Y \mapsto \beta]}[\llbracket G_1 \rrbracket]$ and

$$\begin{aligned} & W^*. \Xi_i \triangleright (\langle E_{i1}[\beta^{E_i^*}/Y], E_{i2}[\beta^{E_i^*}/Y] \rangle t_i[\beta^{E_i^*}/Y] :: \rho'(G_1[\beta/Y])) \\ \mapsto^* & W''. \Xi_i \triangleright (\langle E_{i1}[\beta^{E_i^*}/Y], E_{i2}[\beta^{E_i^*}/Y] \rangle v_{mi} :: \rho'(G_1[\beta/Y])) \\ \mapsto & W''. \Xi_i \triangleright v_{fi} \end{aligned}$$

$$\begin{aligned} & W^*. \Xi_1 \triangleright (\langle E_{11}[\beta^{E_1^*}/Y], E_{12}[\beta^{E_1^*}/Y] \rangle t_1[\beta^{E_1^*}/Y] :: \rho'(G_1[\beta/Y])) \\ \mapsto^{k^*} & W''. \Xi_1 \triangleright (\langle E_{11}[\beta^{E_1^*}/Y], E_{12}[\beta^{E_1^*}/Y] \rangle v_{m1} :: \rho'(G_1[\beta/Y])) \\ \mapsto^m & W''. \Xi_1 \triangleright v_{f1} \end{aligned}$$

Suppose that $v_{fi} = (\varepsilon_{fi} \circ \langle E_{i1}[\beta^{E_i^*}/Y], E_{i2}[\beta^{E_i^*}/Y] \rangle) u_{fi} :: \rho'(G_1[\beta/Y])$. As $E_{12}[\beta^{E_1^*}/Y] \equiv E_{22}[\beta^{E_2^*}/Y]$, then $\text{unlift}(E_{12}[\beta^{E_1^*}/Y]) = \text{unlift}(E_{22}[\beta^{E_2^*}/Y])$. Then we use induction hypothesis using $\rho'[Y \mapsto \beta]$, evidences $\langle E_i''[E_i^*/Y], E_i''[E_i^*/Y] \rangle$, where $E_i''[E_i^*/Y] = \text{lift}_{W'', \Xi_i}(\text{unlift}(E_{i2}[\beta^{E_i^*}/Y]))$ as $E_i = \forall Y. E_i''$,

$$\mathcal{J}(\cdot, \cdot) \text{lift}_{W'', \Xi_i}(G_1[\beta/Y]), \text{lift}_{W'', \Xi_i}(G_1[\beta/Y]) = \langle E_i''[E_i^*/Y], E_i''[E_i^*/Y] \rangle$$

also we know that:

$$\langle E_i''[E_i^*/Y][\alpha^{E_i^*}/X], E_i''[E_i^*/Y][E_i^*/X] \rangle = \langle E_i''[\alpha^{E_i^*}/X][E_i^*/Y], E_i''[E_i^*/X][E_i^*/Y] \rangle$$

Note that $\rho(G_1[\beta/Y]) = \rho[Y \mapsto \beta](G_1)$. Then we know that

$$\begin{aligned} & (\downarrow_k W'', ((\varepsilon_{f1} \circ \langle E_{11}[\beta^{E_1^*}/Y], E_{12}[\beta^{E_1^*}/Y] \rangle) \circ \langle E_1''[\alpha^{E_1^*}/X][E_1^*/Y], E_1''[E_1^*/X][E_1^*/Y] \rangle) u_{f1} :: \rho[Y \mapsto \beta](G_1[G'/X]), \\ & ((\varepsilon_{f2} \circ \langle E_{21}[\beta^{E_2^*}/Y], E_{22}[\beta^{E_2^*}/Y] \rangle) \circ \langle E_2''[\alpha^{E_2^*}/X][E_2^*/Y], E_2''[E_2^*/X][E_2^*/Y] \rangle) u_{f2} :: \rho[Y \mapsto \beta](G_1[G'/X])) \\ & \in \mathcal{V}_{\rho[Y \mapsto \beta]}[\llbracket G_1[G'/X] \rrbracket] \end{aligned}$$

then by inspection of the reduction rules:

$$\begin{aligned} & W^*. \Xi_i \triangleright t''_i \\ \mapsto^* & W''. \Xi_i \triangleright (((\langle E_{i1}[\beta^{E_i^*}/Y], E_{i2}[\beta^{E_i^*}/Y] \rangle) \circ \langle E_i''[\alpha^{E_i^*}/X][\beta^{E_i^*}/Y], E_i''[E_i^*/X][\beta^{E_i^*}/Y] \rangle) v_{mi} :: \rho'(G_1[\beta/Y])) \\ \mapsto & W''. \Xi_i \triangleright (\varepsilon_{fi} \circ (\langle E_{i1}[\beta^{E_i^*}/Y], E_{i2}[\beta^{E_i^*}/Y] \rangle \circ \langle E_i''[\alpha^{E_i^*}/X][E_i^*/Y], E_i''[E_i^*/X][E_i^*/Y] \rangle)) u_{fi} :: \rho[Y \mapsto \beta](G_i[G'/X]) \end{aligned}$$

and by Lemma A.103, we know that those two values belong to the interpretation of $\mathcal{V}_{\rho[Y \mapsto \beta]}[\llbracket G_1[G'/X] \rrbracket]$, and the result holds. Note that $\downarrow_k W''.k + m + k^* = W^*$.

Case (Pair Type: $G_1 \times G_2$). Analogous to the function case.

Case (Base Type: B). Trivial.

Then we prove as (2):

Case (Type Variable X: $G = X$). Let $v_i = \langle H_{i1}, E_{i2} \rangle u_i :: X[G'/X] = \langle H_{i1}, E_{i2} \rangle u_i :: G'$. Then we know that

$$(W, \langle H_{11}, E_{12} \rangle u_1 :: G', \langle H_{21}, E_{22} \rangle u_2 :: G') \in \mathcal{V}_\rho[G']$$

and $\varepsilon_i^{-1} = \langle E'_i, \alpha^{E'_i} \rangle$. Then we have to prove that

$$(\downarrow_l W, (\langle H_{11}, E_{12} \rangle \circ \langle E'_1, \alpha^{E'_1} \rangle) u_1 :: \alpha, (\langle H_{21}, E_{22} \rangle \circ \langle E'_2, \alpha^{E'_2} \rangle) u_2 :: \alpha) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[\alpha]$$

By Lemma A.96, we assume that both combinations of evidence are defined (otherwise the result holds immediately). Then by definition of transitivity and Lemma A.115, we know that $(\langle H_{i1}, E_{i2} \rangle \circ \langle E'_i, \alpha^{E'_i} \rangle) = \langle H_{i1}, \alpha^{E_{i2}} \rangle$. Then we have to prove that

$$(\downarrow_l W, \langle H_{11}, \alpha^{E_{12}} \rangle u_1 :: \alpha, \langle H_{21}, \alpha^{E_{22}} \rangle u_2 :: \alpha) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[\alpha]$$

but as α is sync, then that is equivalent to

$$(\downarrow_{l-1} W, \langle H_{11}, E_{12} \rangle u_1 :: G', \langle H_{21}, E_{22} \rangle u_2 :: G') \in \mathcal{V}_\rho[G']$$

which follows by the premise and Lemma A.97.

Also, we have to prove that $(\forall \Xi', \varepsilon', G^*$ such that $(\downarrow_{l-1} W \in \mathcal{S}[\Xi'] \wedge \varepsilon' \vdash \Xi' \vdash \alpha \sim G^*)$, we get that

$$(\downarrow_{l-1} W, \varepsilon'(\langle H_{11}, \alpha^{E_{12}} \rangle u_1 :: \alpha) :: G^*, \varepsilon'(\langle H_{21}, \alpha^{E_{22}} \rangle u_2 :: \alpha) :: G^*) \in \mathcal{I}_\rho[G^*]$$

or what is the same $((\langle H_{11}, \alpha^{E_{12}} \rangle \circ \varepsilon')$ fails the result follows immediately)

$$(\downarrow_{l-1-k'} W, (\langle H_{11}, \alpha^{E_{12}} \rangle \circ \varepsilon') u_1 :: G^*, (\langle H_{21}, \alpha^{E_{22}} \rangle \circ \varepsilon') u_2 :: G^*) \in \mathcal{V}_\rho[G^*]$$

where $\varepsilon' = \langle \alpha^{E_1^*}, E_2^* \rangle$ and $\varepsilon'.n = k'$. By definition of transitivity and Lemma A.115, we know that

$$\langle H_{i1}, \alpha^{E_{i2}} \rangle \circ \langle \alpha^{E_1^*}, E_2^* \rangle = \langle H_{i1}, E_{i2} \rangle \circ \langle E_1^*, E_2^* \rangle$$

We know that $\langle E_1^*, E_2^* \rangle \vdash \Xi' \vdash G' \sim G^*$. Since $\langle E_1^*, E_2^* \rangle \vdash \Xi \vdash G' \sim G^*$, $\downarrow_{l-1} W \in \mathcal{S}[\Xi']$, $(\downarrow_{l-1} W, \langle H_{11}, E_{12} \rangle u_1 :: G', \langle H_{21}, E_{22} \rangle u_2 :: G') \in \mathcal{V}_\rho[G']$, by Lemma A.101, we know that (since $(\langle H_{11}, \alpha^{E_{12}} \rangle \circ \varepsilon')$ does not fail then $(\langle H_{11}, E_{12} \rangle \circ \langle E_1^*, E_2^* \rangle)$ also does not fail by the transitivity rules)

$$(\downarrow_{l-1-k'} W, (\langle H_{11}, E_{12} \rangle \circ \langle E_1^*, E_2^* \rangle) u_1 :: G^*, (\langle H_{21}, E_{22} \rangle \circ \langle E_1^*, E_2^* \rangle) u_2 :: G^*) \in \mathcal{V}_\rho[G^*]$$

The result follows immediately.

Case (Type Variable Y: $G = Y$). Let $v_i = \langle H_{i1}, \beta^{E_{i2}} \rangle u_i :: \rho(Y[G'/X]) = \langle H_{i1}, \beta^{E_{i2}} \rangle u_i :: \beta$ (where $\rho(Y) = \beta$). Then we know that

$$(W, \langle H_{11}, \beta^{E_{12}} \rangle u_1 :: \beta, \langle H_{21}, \beta^{E_{22}} \rangle u_2 :: \beta) \in \mathcal{V}_\rho[\beta]$$

We know that $\varepsilon_i^{-1} \vdash W.\Xi_i \vdash \beta \sim \beta$, therefore ε_i^{-1} has to have the form $\varepsilon_i^{-1} = \langle \beta^{E'_i}, \beta^{E'_i} \rangle = \mathcal{G}(\cdot, \text{lift}_{W.\Xi_i}(\beta), \text{lift}_{W.\Xi_i}(\beta))$. As ε_i^{-1} is the initial evidence for β , then $E_{i2} \sqsubseteq E'_i$, and therefore by definition of the transitivity and Lemma A.115:

$$\langle H_{i1}, \beta^{E_{i2}} \rangle \circ \langle \beta^{E'_i}, \beta^{E'_i} \rangle = \langle H_{i1}, \beta^{E_{i2}} \rangle$$

Then we have to prove that:

$$(\downarrow_l W, (\langle H_{11}, \beta^{E_{12}} \rangle \circ \langle \beta^{E_1}, \beta^{E_1} \rangle) u_1 :: \beta, (\langle H_{21}, \beta^{E_{22}} \rangle \circ \langle E_2', \beta^{E_2'} \rangle) u_2 :: \beta) \in \mathcal{V}_\rho[X \mapsto \alpha][[\beta]]$$

or what is the same

$$(\downarrow_l W, \langle H_{11}, \beta^{E_{12}} \rangle u_1 :: \beta, \langle H_{21}, \beta^{E_{22}} \rangle u_2 :: \beta) \in \mathcal{V}_\rho[[\beta]]$$

which follows by the premise and Lemma A.97.

Case (Unknown Type: $G = ?$). Let $v_i = \langle H_{i1}, E_{i2} \rangle u_i :: ?$. Then by definition of $\mathcal{V}_\rho[[?]]$, let $G'' = \text{const}(E_{i2})$ (where $G'' \neq ?$). Then we know

$$(W, \langle H_{11}, E_{12} \rangle u_1 :: G'', \langle H_{21}, E_{22} \rangle u_2 :: G'') \in \mathcal{V}_\rho[[G'']]$$

If $\varepsilon_i^{-1} = \langle ?, ? \rangle$, then, $\langle H_{i1}, E_{i2} \rangle \circ \langle ?, ? \rangle = \langle H_{i1}, E_{i2} \rangle$, by Lemma A.115, the result holds immediately.

If $\varepsilon_i^{-1} \neq \langle ?, ? \rangle$. Then we proceed similar to the other cases where $G \neq ?$. Note that we know that

$$(W, \langle H_{11}, E_{12} \rangle u_1 :: G'', \langle H_{21}, E_{22} \rangle u_2 :: G'') \in \mathcal{V}_\rho[[G'']]$$

where $G'' \neq ?$ and we are required to prove that

$$(W, \varepsilon_1(\langle H_{11}, E_{12} \rangle u_1 :: G'') :: G'', \varepsilon_2(\langle H_{21}, E_{22} \rangle u_2 :: G'')) \in \mathcal{V}_\rho[[G'']]$$

Case (Function Type: $G = G_1 \rightarrow G_2$). Let $v_i = \varepsilon_i'(\lambda x : G_i'.t_i) :: \rho(G[G'/X])$. We know that

$$(W, v_1, v_2) \in \mathcal{V}_\rho[[G_1[G'/X] \rightarrow G_2[G'/X]]]$$

Then we have to prove that

$$\begin{aligned} &(\downarrow_l W, (\varepsilon_1' \circ \varepsilon_1^{-1})(\lambda x : G_1'.t_1) :: \rho'(G_1) \rightarrow \rho'(G_2), \\ &(\varepsilon_2' \circ \varepsilon_2^{-1})(\lambda x : G_2'.t_2) :: \rho'(G_1) \rightarrow \rho'(G_2)) \in \mathcal{V}_{\rho'}[[G_1 \rightarrow G_2]] \end{aligned}$$

Let us call $v_i'' = (\varepsilon_i' \circ \varepsilon_i^{-1})(\lambda x : G_i'.t_i) :: \rho'(G_1) \rightarrow \rho'(G_2)$. By unfolding, we have to prove that

$$\forall W' \succeq (\downarrow_l W). \forall v_1', v_2'. (\downarrow_l W', v_1', v_2') \in \mathcal{V}_{\rho'}[[G_1]] \Rightarrow (W', v_1'' v_1', v_2'' v_2') \in \mathcal{F}_{\rho'}[[G_2]]$$

Suppose that $v_i' = \varepsilon_i'' u_i' :: \rho'(G_1)$, by inspection of the reduction rules, we know that

$$W'. \Xi_i \triangleright v_i'' v_i' \mapsto^* W'. \Xi_i \triangleright (\text{cod}(\varepsilon_i') \circ \text{cod}(\varepsilon_i^{-1})) t_i [(\varepsilon_i'' \circ (\text{dom}(\varepsilon_i^{-1}) \circ \text{dom}(\varepsilon_i'))) u_i' :: G_i'] / x :: \rho'(G_2)$$

This is equivalent by Lemma A.103,

$$W'. \Xi_i \triangleright v_i'' v_i' \mapsto^* W'. \Xi_i \triangleright (\text{cod}(\varepsilon_i') \circ \text{cod}(\varepsilon_i^{-1})) t_i [((\varepsilon_i'' \circ \text{dom}(\varepsilon_i^{-1})) \circ \text{dom}(\varepsilon_i')) u_i' :: G_i'] / x :: \rho'(G_2)$$

Also, we know that

$$\begin{aligned} &W'. \Xi_1 \triangleright v_1'' v_1' \mapsto^{l+m+1} \\ &W'. \Xi_1 \triangleright (\text{cod}(\varepsilon_1') \circ \text{cod}(\varepsilon_1^{-1})) t_1 [((\varepsilon_1'' \circ \text{dom}(\varepsilon_1^{-1})) \circ \text{dom}(\varepsilon_1')) u_1' :: G_1'] / x :: \rho'(G_2) \mapsto^{k^*} \\ &\Xi_1 \triangleright (\text{cod}(\varepsilon_1') \circ \text{cod}(\varepsilon_1^{-1})) v_{1f} :: \rho'(G_2) \mapsto^{l+m} \end{aligned}$$

$$\Xi_1 \triangleright v_1^*$$

where $v_{1f} = \varepsilon_{1f} u_{1f} :: \rho(G_2[G'/X])$ and $v_1^* = (\varepsilon_{1f} \mathbin{\text{;}} \text{cod}(\varepsilon'_1) \mathbin{\text{;}} \text{cod}(\varepsilon_1^{-1})) u_{1f} :: \rho'(G_2)$.

Notice that $\text{dom}(\varepsilon_i^{-1}) \vdash W.\Xi_i \vdash \rho(G_1[\alpha/X]) \sim \rho(G_1[G'/X])$, and as $\text{dom}(\varepsilon_i^{-1})$ is constructed using the interior (and thus $\pi_2(\varepsilon_i'') \sqsubseteq \pi_1(\text{dom}(\varepsilon_i^{-1}))$), then by definition of evidence $(\varepsilon_i'' \mathbin{\text{;}} \text{dom}(\varepsilon_i^{-1}))$ is always defined. We can use induction hypothesis on v'_i , with evidences $\text{dom}(\varepsilon_i^{-1})$.

Then we know that

$$(\downarrow_{l+1} W', (\varepsilon_1'' \mathbin{\text{;}} \text{dom}(\varepsilon_1^{-1})) u'_1 :: \rho(G_1[G'/X]), (\varepsilon_2'' \mathbin{\text{;}} \text{dom}(\varepsilon_2^{-1})) u'_2 :: \rho(G_1[G'/X])) \in \mathcal{V}_\rho[[G_1[G'/X]]]$$

Let us call $v_i''' = (\varepsilon_i'' \mathbin{\text{;}} \text{dom}(\varepsilon_i^{-1})) u'_i :: \rho(G_1[G'/X])$.

Now we instantiate

$$(W, v_1, v_2) \in \mathcal{V}_\rho[[G_1[G'/X] \rightarrow G_2[G'/X]]]$$

with $(\downarrow_l W')$ and v_i''' , to obtain that either both executions reduce to an error (then the result holds immediately), or $\exists W'' \succeq (\downarrow_l W')$ such that $(W'', v'_{f1}, v'_{f2}) \in \mathcal{V}_\rho[[G_2[G'/X]]]$, $W''.j + 2m + k^* = (\downarrow_l W').j$ ($W''.j + 1 + l + 2m + k^* = W'.j$) and

$$\begin{aligned} W'.\Xi_i \triangleright v_i v_i''' &\mapsto W'.\Xi_i \triangleright \text{cod}(\varepsilon'_i) t_i [((\varepsilon_i'' \mathbin{\text{;}} \text{dom}(\varepsilon_i^{-1})) \mathbin{\text{;}} \text{dom}(\varepsilon'_i)) u'_i :: G'_i/x] :: \rho(G_2[G'/X]) \\ &\mapsto^* W''.\Xi_i \triangleright v'_{fi} \end{aligned}$$

Therefore, we know that

$$\begin{aligned} W'.\Xi_1 \triangleright v_1 v_1''' &\mapsto^{m+1} \\ W'.\Xi_1 \triangleright \text{cod}(\varepsilon'_1) t_1 [((\varepsilon_1'' \mathbin{\text{;}} \text{dom}(\varepsilon_1^{-1})) \mathbin{\text{;}} \text{dom}(\varepsilon'_1)) u'_1 :: G'_1/x] :: \rho(G_2[G'/X]) &\mapsto^{k^*} \\ \Xi_1 \triangleright \text{cod}(\varepsilon'_1) v_{f1} :: \rho(G_2[G'/X]) &\mapsto^m \\ W''.\Xi_1 \triangleright v'_{f1} & \end{aligned}$$

Suppose that $v'_{fi} = \varepsilon'_{fi} u_{fi} :: \rho(G_2[G'/X])$ and $\varepsilon'_{f1} = \varepsilon_{f1} \mathbin{\text{;}} \text{cod}(\varepsilon'_1)$. Then we use induction hypothesis once again using evidences $\text{cod}(\varepsilon_i^{-1})$ and $(W'', v'_{f1}, v'_{f2}) \in \mathcal{V}_\rho[[G_2[G'/X]]]$, (noticing that the combination of evidence does not fail as the evidence is obtained via the interior function i.e. the less precise evidence possible), to obtain that,

$$(\downarrow_l W'', (\varepsilon_{f1} \mathbin{\text{;}} \text{cod}(\varepsilon'_1) \mathbin{\text{;}} \text{cod}(\varepsilon_1^{-1})) u_{f1} :: \rho'(G_2), (\varepsilon_{f2} \mathbin{\text{;}} \text{cod}(\varepsilon'_2) \mathbin{\text{;}} \text{cod}(\varepsilon_2^{-1})) u_{f2} :: \rho'(G_2)) \in \mathcal{V}_{\rho'}[[G_2]]$$

Note that $(\downarrow_l W'').j + 1 + 2l + 2m + k^* = W'.j$, and the result holds.

The remaining cases are similar.

□

Lemma A.99 (Compositionality) *If*

- $W.\Xi_i(\alpha) = \rho(G')$ and $W.\kappa(\alpha) = \mathcal{V}_\rho[[G']]$,
- $E'_i = \text{lift}_{W.\Xi_i}(\rho(G'))$,

- $E_i = \text{lift}_{W, \Xi_i}(G_p)$ for some $G_p \sqsubseteq \rho(G)$,
- $\rho' = \rho[X \mapsto \alpha]$,
- $\varepsilon_i = \langle E_i[\alpha^{E'_i}/X], E_i[E'_i/X] \rangle$, such that $\varepsilon_i \vdash W, \Xi_i \vdash \rho(G[\alpha/X]) \sim \rho(G[G'/X])$, and
- $\varepsilon_i^{-1} = \langle E_i[E'_i/X], E_i[\alpha^{E'_i}/X] \rangle$, such that $\varepsilon_i^{-1} \vdash W, \Xi_i \vdash \rho(G[G'/X]) \sim \rho(G[\alpha/X])$, then

1. $(W, v_1, v_2) \in \mathcal{V}_\rho[[G]] \Rightarrow (W, \varepsilon_1 v_1 :: \rho(G[G'/X]), \varepsilon_2 v_2 :: \rho(G[G'/X])) \in \mathcal{J}_\rho[[G[G'/X]]]$
2. $(W, v_1, v_2) \in \mathcal{V}_\rho[[G[G'/X]]] \Rightarrow (W, \varepsilon_1^{-1} v_1 :: \rho'(G), \varepsilon_2^{-1} v_2 :: \rho'(G)) \in \mathcal{J}_{\rho'}[[G]]$

PROOF. Direct by Prop. 3.31. □

Definition A.100 $\rho \vdash \varepsilon_1 \equiv \varepsilon_2$ if $\text{unlift}(\pi_2(\varepsilon_1)) = \text{unlift}(\pi_2(\varepsilon_2))$

Proposition A.101 *If*

- $(W, v_1, v_2) \in \mathcal{V}_\rho[[G]]$
- $\varepsilon \Vdash \Xi; \Delta \vdash G \sim G'$
- $W \in \mathcal{S}[[\Xi]]$ and $(W, \rho) \in \mathcal{D}[[\Delta]]$
- $\forall \alpha \in \text{dom}(\Xi). \text{sync}(\alpha, W)$

then:

$$(W, \rho_1(\varepsilon)v_1 :: \rho(G'), \rho_2(\varepsilon)v_2 :: \rho(G')) \in \mathcal{J}_\rho[[G']]$$

$$\text{where } \text{sync}(\alpha, W) \iff W, \Xi_1(\alpha) = W, \Xi_2(\alpha) \wedge W, \kappa(\alpha) = \lfloor \mathcal{V}_\emptyset[[W, \Xi_i(\alpha)]] \rfloor_{W, j}.$$

PROOF. We proceed by induction on G and W, j . We know that $u_i \in G_i$ for some G_i , notice that $G_i \in \text{HEADTYPE} \cup \text{TYPEVAR}$. In every case we apply Lemma A.111 to show that $(\varepsilon_1 \circledast \varepsilon_1^\rho) \iff (\varepsilon_2 \circledast \varepsilon_2^\rho)$, so in all cases we assume that the transitivity does not fail (otherwise the proof holds immediately). Let us call $\varepsilon_1^\rho = \rho_1(\varepsilon)$ and $\varepsilon_2^\rho = \rho_2(\varepsilon)$. Let's suppose that $\varepsilon_1^\rho.n = k$ and $\varepsilon_1.n = l$.

Case (Base type: $G = B$ and $G' = B$). We know that v_i has the form $\langle B, B \rangle u :: B$, and we know that $(W, \langle B, B \rangle u :: B, \langle B, B \rangle u :: B) \in \mathcal{V}_\rho[[B]]$. Also as $\varepsilon \vdash \Xi; \Delta \vdash B \sim B$, then $\varepsilon = \langle B, B \rangle$, then as $\rho_i(B) = B$, $\varepsilon_i \circledast \rho_i(\varepsilon) = \varepsilon_i$, and we have to prove that $(\downarrow_k W, \langle B, B \rangle u :: B, \langle B, B \rangle u :: B) \in \mathcal{V}_\rho[[B]]$, which follows immediately because the premise and Lemma A.97.

Case (Function type: $G = G_1'' \rightarrow G_2''$, and $G' = G_1' \rightarrow G_2'$). We know that:

$$(W, v_1, v_2) \in \mathcal{V}_\rho[[G_1'' \rightarrow G_2'']]$$

Where $v_i = \varepsilon_i(\lambda x : G_{1i}.t_i) :: \rho(G_1'' \rightarrow G_2'')$ and $\varepsilon_i \vdash W, \Xi_i \vdash G_i \sim \rho(G_1' \rightarrow G_2')$.

We have to prove that:

$$(W, \varepsilon_1^\rho v_1 :: \rho(G_1' \rightarrow G_2'), \varepsilon_2^\rho v_2 :: \rho(G_1' \rightarrow G_2')) \in \mathcal{J}_\rho[[G_1' \rightarrow G_2']]$$

Or what is the same:

$$(\downarrow_l W, (\varepsilon_1 \circledast \varepsilon_1^\rho)(\lambda x : G_{11}.t_1) :: \rho(G_1' \rightarrow G_2'), (\varepsilon_2 \circledast \varepsilon_2^\rho)(\lambda x : G_{12}.t_2) :: \rho(G_1' \rightarrow G_2')) \in \mathcal{J}_\rho[[G_1' \rightarrow G_2']]$$

First we suppose that $(\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho)$ does not fail and $(\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho).n = k + l$, then we have to prove that:

$$\forall W' \succeq \downarrow_1 W. \forall v'_1, v'_2. (\downarrow_1 W', v'_1, v'_2) \in \mathcal{V}_\rho \llbracket G'_1 \rrbracket \Rightarrow$$

$$(W', [(\varepsilon_1 \mathbin{\text{;}} \varepsilon_1^\rho)(\lambda x : G_{11}.t_1) :: \rho(G'_1 \rightarrow G'_2)] v'_1, [(\varepsilon_2 \mathbin{\text{;}} \varepsilon_2^\rho)(\lambda x : G_{12}.t_2) :: \rho(G'_1 \rightarrow G'_2)] v'_2) \in \mathcal{J}_\rho \llbracket G'_2 \rrbracket$$

where $v'_i = \varepsilon'_i u'_i :: \rho(G'_1)$. Note that by the reduction rule of application terms, we obtain that:

$$\begin{aligned} W'.\Xi_i \triangleright ((\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho)(\lambda x : G_{1i}.t_i) :: \rho(G'_1 \rightarrow G'_2)) (\varepsilon'_i u'_i :: \rho(G'_1)) &\longrightarrow^* \\ W'.\Xi_i \triangleright \text{cod}(\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho) ([(\varepsilon'_i \mathbin{\text{;}} \text{dom}(\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho)) u'_i :: G_{1i}] / x] t_i) :: \rho(G'_2) \end{aligned}$$

We know by the Proposition A.105 that $\text{dom}(\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho) = \text{dom}(\varepsilon_i^\rho) \mathbin{\text{;}} \text{dom}(\varepsilon_i)$. Then by the Proposition A.103 we know that:

$$\varepsilon'_i \mathbin{\text{;}} (\text{dom}(\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho)) = \varepsilon'_i \mathbin{\text{;}} (\text{dom}(\varepsilon_i^\rho) \mathbin{\text{;}} \text{dom}(\varepsilon_i)) = (\varepsilon'_i \mathbin{\text{;}} \text{dom}(\varepsilon_i^\rho)) \mathbin{\text{;}} \text{dom}(\varepsilon_i)$$

Also, by the Proposition A.106 it follows that: $\text{cod}(\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho) = \text{cod}(\varepsilon_i) \mathbin{\text{;}} \text{cod}(\varepsilon_i^\rho)$.

Then the following result is true:

$$\begin{aligned} W'.\Xi_i \triangleright \text{cod}(\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho) ([(\varepsilon'_i \mathbin{\text{;}} \text{dom}(\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho)) u'_i :: G_{1i}] / x] t_i) :: \rho(G'_2) &= \\ W'.\Xi_i \triangleright \text{cod}((\varepsilon_i \mathbin{\text{;}} \text{cod}(\varepsilon_i^\rho)) ([(\varepsilon'_i \mathbin{\text{;}} \text{dom}(\varepsilon_i^\rho)) \mathbin{\text{;}} \text{dom}(\varepsilon_i)] u'_i :: G_{1i}] / x] t_i) :: \rho(G'_2) \end{aligned}$$

So, we know that:

$$\begin{aligned} W'.\Xi_1 \triangleright ((\varepsilon_1 \mathbin{\text{;}} \varepsilon_1^\rho)(\lambda x : G_{11}.t_1) :: \rho(G'_1 \rightarrow G'_2)) (\varepsilon'_1 u'_1 :: \rho(G'_1)) &\longrightarrow^{l+k+1} \\ W'.\Xi_1 \triangleright \text{cod}(\varepsilon_1 \mathbin{\text{;}} \varepsilon_1^\rho) ([(\varepsilon'_1 \mathbin{\text{;}} \text{dom}(\varepsilon_1 \mathbin{\text{;}} \varepsilon_1^\rho)) u'_1 :: G_{11}] / x] t_1) :: \rho(G'_2) &= \\ W'.\Xi_1 \triangleright \text{cod}((\varepsilon_1 \mathbin{\text{;}} \text{cod}(\varepsilon_1^\rho)) ([(\varepsilon'_1 \mathbin{\text{;}} \text{dom}(\varepsilon_1^\rho)) \mathbin{\text{;}} \text{dom}(\varepsilon_1)] u'_1 :: G_{11}] / x] t_1) :: \rho(G'_2) &\longrightarrow^{k*} \\ \Xi_1 \triangleright (\text{cod}(\varepsilon_1) \mathbin{\text{;}} \text{cod}(\varepsilon_1^\rho)) v_1^* :: \rho(G'_2) &\longrightarrow^{l+k} \\ \Xi_1 \triangleright (\varepsilon''_1 \mathbin{\text{;}} (\text{cod}(\varepsilon_1) \mathbin{\text{;}} \text{cod}(\varepsilon_1^\rho))) u_{1f} :: \rho(G'_2) \end{aligned}$$

where $v_1^* = \varepsilon''_1 u_{1f} :: \rho(G'_2)$ and $v_{1f} = (\varepsilon''_1 \mathbin{\text{;}} (\text{cod}(\varepsilon_1) \mathbin{\text{;}} \text{cod}(\varepsilon_1^\rho))) u_{1f} :: \rho(G'_2)$.

We instantiate the induction hypothesis in $(\downarrow_1 W', v'_1, v'_2) \in \mathcal{V}_\rho \llbracket G'_1 \rrbracket$ with the type G''_1 and the evidences $\text{dom}(\varepsilon) \vdash \Xi; \Delta \vdash G'_1 \sim G''_1$, where $\text{dom}(\varepsilon).n = l$. We obtain that:

$$(\downarrow_1 W', \text{dom}(\varepsilon_1^\rho) v'_1 :: G''_1, \text{dom}(\varepsilon_2^\rho) v'_2 :: G''_1) \in \mathcal{J}_\rho \llbracket G''_1 \rrbracket$$

In particular we focus on a pair of values such that $(\varepsilon'_i \mathbin{\text{;}} \text{dom}(\varepsilon_i^\rho))$ does not fail (otherwise the result follows immediately). Then it is true that:

$$(\downarrow_{l+1} W', (\varepsilon'_1 \mathbin{\text{;}} \text{dom}(\varepsilon_1^\rho)) u'_1 :: G''_1, (\varepsilon'_2 \mathbin{\text{;}} \text{dom}(\varepsilon_2^\rho)) u'_2 :: G''_1) \in \mathcal{V}_\rho \llbracket G''_1 \rrbracket$$

By the definition of $\mathcal{V}_\rho \llbracket G''_1 \rightarrow G''_2 \rrbracket$ we know that:

$$\forall W'' \succeq W. \forall v''_1, v''_2. (\downarrow_1 W'', v''_1, v''_2) \in \mathcal{V}_\rho \llbracket G''_1 \rrbracket \Rightarrow (W'', v_1 v''_1, v_2 v''_2) \in \mathcal{J}_\rho \llbracket G''_2 \rrbracket$$

We instantiate $v_i'' = (\varepsilon_i' \circledast \text{dom}(\varepsilon_i^\rho))u_i' :: \rho(G_1'')$ and $W'' = \downarrow_l W'$. Then we obtain that:

$$\begin{aligned} (\downarrow_l W', ((\varepsilon_1(\lambda x : G_{11}.t_1) :: \rho(G_1'' \rightarrow G_2'')) ((\varepsilon_1' \circledast \text{dom}(\varepsilon_1^\rho))u_1' :: \rho(G_1'')), \\ (\varepsilon_2(\lambda x : G_{12}.t_2) :: \rho(G_1'' \rightarrow G_2'')) ((\varepsilon_2' \circledast \text{dom}(\varepsilon_2^\rho))u_2' :: \rho(G_1''))) \in \mathcal{T}_\rho[[G_2'']] \end{aligned}$$

Then by Lemma A.103, as $(\varepsilon_1' \circledast \text{dom}(\varepsilon_1^\rho)) \circledast \text{dom}(\varepsilon_1) = \varepsilon_1' \circledast (\text{dom}(\varepsilon_1^\rho) \circledast \text{dom}(\varepsilon_1))$, then if $(\text{dom}(\varepsilon_1^\rho)) \circledast \text{dom}(\varepsilon_1)$ is not defined and $(\text{dom}(\varepsilon_2^\rho)) \circledast \text{dom}(\varepsilon_2)$ is defined, we get a contradiction as both must behave uniformly as the terms belong to $\mathcal{T}_\rho[[G_2'']]$. Then if both combination of evidence fail, then the result follows immediately. Let us suppose that the combination does not fail, then

$$\begin{aligned} W'.\Xi_i \triangleright (\varepsilon_i(\lambda x : G_{1i}.t_i) :: \rho(G_1'' \rightarrow G_2'')) ((\varepsilon_i' \circledast \text{dom}(\varepsilon_i^\rho))u_i' :: \rho(G_1'')) \longrightarrow^* \\ W'.\Xi_i \triangleright \text{cod}(\varepsilon_i)((\varepsilon_i' \circledast \text{dom}(\varepsilon_i^\rho)) \circledast \text{dom}(\varepsilon_i))u_i' :: G_{1i}/x]t_i :: \rho(G_2') \end{aligned}$$

So, we know that:

$$\begin{aligned} W'.\Xi_1 \triangleright ((\varepsilon_1(\lambda x : G_{11}.t_1) :: \rho(G_1'' \rightarrow G_2'')) ((\varepsilon_1' \circledast \text{dom}(\varepsilon_1^\rho))u_1' :: \rho(G_1'')) \longrightarrow^{k+1} \\ W'.\Xi_1 \triangleright \text{cod}(\varepsilon_1)((\varepsilon_1' \circledast \text{dom}(\varepsilon_1^\rho)) \circledast \text{dom}(\varepsilon_1))u_1' :: G_{11}/x]t_1 :: \rho(G_2') \longrightarrow^{k^*} \\ \Xi_1 \triangleright \text{cod}(\varepsilon_1)v_1^* :: \rho(G_2') \longrightarrow^k \\ \Xi_1 \triangleright (\varepsilon_1'' \circledast \text{cod}(\varepsilon_1))u_{1f} :: \rho(G_2') \end{aligned}$$

where $v_1^* = (\varepsilon_1'' \circledast \text{cod}(\varepsilon_1))u_{1f} :: \rho(G_2')$.

Thus, we know that $\exists W''' \succeq \downarrow_l W'$ such that $(W''', v_1^*, v_2^*) \in \mathcal{V}_\rho[[G_2'']]$, $W'''.\Xi_1 = \Xi_1$ and $W'''.j + 1 + 2k + k^* = (\downarrow_l W').j$, or what is the same $W'''.j + 1 + 2k + k^* + l = W'.j$. Then, we know that

$$W'.\Xi_i \triangleright \text{cod}(\varepsilon_i)((\varepsilon_i' \circledast \text{dom}(\varepsilon_i^\rho)) \circledast \text{dom}(\varepsilon_i))u_i' :: G_{1i}/x]t_i :: \rho(G_2') \longrightarrow^* W'''.\Xi_i \triangleright v_i^*$$

We instantiate the induction hypothesis in the previous result $((W''', v_1^*, v_2^*) \in \mathcal{V}_\rho[[G_2'']])$ with the type G_2' and the evidence $\text{cod}(\varepsilon) \vdash \Xi; \Delta \vdash G_2'' \sim G_2'$, where $\text{cod}(\varepsilon_1^\rho).n = l$, then we obtain that:

$$(W''', \text{cod}(\varepsilon_1^\rho)v_1^* :: \rho(G_2'), \text{cod}(\varepsilon_2^\rho)v_2^* :: \rho(G_2'))' \in \mathcal{T}_\rho[[G_2']]$$

Then v_i^* has to have the form: $v_i^* = (\varepsilon_i'' \circledast \text{cod}(\varepsilon_i))u_{if} :: \rho(G_2')$ form some ε_i'' , u_{if} . Then as $(\varepsilon_1'' \circledast \text{cod}(\varepsilon_1)) \circledast \text{cod}(\varepsilon_1^\rho) = \varepsilon_1'' \circledast (\text{cod}(\varepsilon_1) \circledast \text{cod}(\varepsilon_1^\rho))$, then $(\text{cod}(\varepsilon_1) \circledast \text{cod}(\varepsilon_1^\rho))$ must behave uniformly (either the two of them fail, or the two of them does not fail). Thus, we get that $(\downarrow_l W''', v_{1f}, v_{2f}) \in \mathcal{V}_\rho[[G_2']]$ where $v_{if} = (\varepsilon_i'' \circledast (\text{cod}(\varepsilon_i) \circledast \text{cod}(\varepsilon_i^\rho)))u_{if} :: \rho(G_2')$ and $W'''.j + 1 + 2k + 2l + k = W'.j$. Therefore, the result immediately.

Case (Universal Type: $G = \forall X.G_1''$ and $G' = \forall X.G_1'$). We know that:

$$(W, v_1, v_2) \in \mathcal{V}_\rho[[\forall X.G_1'']]$$

Where $v_i = \varepsilon_i(\Lambda X.t_i) :: \forall X.\rho(G_1'')$ and $\varepsilon_i \vdash W.\Xi_i \vdash G_i \sim \forall X.\rho(G_1'')$.

We have to prove that:

$$(W, \varepsilon_1^\rho v_1 :: \forall X. \rho(G'_1), \varepsilon_2^\rho v_2 :: \forall X. \rho(G'_1)) \in \mathcal{J}_\rho[\forall X. G'_1]$$

As $(\varepsilon_i \circ \varepsilon_i^\rho)$ does not fail, then by the definition of $\mathcal{J}_\rho[\forall X. G'_1]$ we have to prove that:

$$(\downarrow_k W, (\varepsilon_1 \circ \varepsilon_1^\rho)(\Lambda X. t_1) :: \forall X. \rho(G'_1), (\varepsilon_2 \circ \varepsilon_2^\rho)(\Lambda X. t_2) :: \forall X. \rho(G'_1)) \in \mathcal{V}_\rho[\forall X. G'_1]$$

or what is the same:

$$\begin{aligned} & \forall W'' \succeq (\downarrow_k W). \forall t'_1, t'_2, G_1^*, G_2^*, \alpha, \varepsilon_{11}, \varepsilon_{21}. \forall R \in \text{REL}_{W''}^j[G_1^*, G_2^*]. \\ & (W''. \Xi_1 \vdash G_1^* \wedge W''. \Xi_2 \vdash G_2^* \wedge \\ & W''. \Xi_1 \triangleright ((\varepsilon_1 \circ \varepsilon_1^\rho) u_1 :: \forall X. G'_1)[G_1^*] \longrightarrow W''. \Xi_1, \alpha := G_1^* \triangleright \varepsilon_{11} t'_1 :: G'_1[G_1^*/X] \wedge \\ & W''. \Xi_2 \triangleright ((\varepsilon_2 \circ \varepsilon_2^\rho) u_2 :: \forall X. G'_1)[G_2^*] \longrightarrow W''. \Xi_2, \alpha := G_2^* \triangleright \varepsilon_{21} t'_2 :: G'_1[G_2^*/X]) \Rightarrow \\ & (W''', t'_1, t'_2) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[G'_1] \end{aligned}$$

where $W''' = \downarrow(W'' \boxtimes (\alpha, G_1^*, G_2^*, R))$. Note that by the reduction rule of type application, we obtain that:

$$\begin{aligned} & W''. \Xi_i \triangleright ((\varepsilon_i \circ \varepsilon_i^\rho) \Lambda X. t_i :: \forall X. \rho(G'_1)) [G_i^*] \longrightarrow \\ & W''. \Xi_i, \alpha := G_i^* \triangleright \varepsilon_{\forall X. \rho(G'_1)}^{E_i/\alpha^{E_i}} ((\varepsilon_i \circ \varepsilon_i^\rho) [\alpha^{E_i}] t_i [\alpha^{E_i}/X] :: \rho(G'_1) [\alpha/X]) :: \rho(G'_1) [G_i^*/X] \end{aligned}$$

where $E_i = \text{lift}_{(W'', \Xi_i)}(G_i^*)$. The resulting evidences $\varepsilon_i \circ \varepsilon_i^\rho$ have the form: $\langle \forall X. E_{i1}, \forall X. E_{i2} \rangle$, then:

$$\begin{aligned} & \varepsilon_{\forall X. \rho(G'_1)}^{E_i/\alpha^{E_i}} ((\varepsilon_i \circ \varepsilon_i^\rho) [\alpha^{E_i}] t_i [\alpha^{E_i}/X] :: \rho(G'_1) [\alpha/X]) :: \rho(G'_1) [G_i^*/X] = \\ & \varepsilon_{\forall X. \rho(G'_1)}^{E_i/\alpha^{E_i}} (\langle E_{i1} [\alpha^{E_i}/X], E_{i2} [\alpha^{E_i}/X] \rangle t_i [\alpha^{E_i}/X] :: \rho(G'_1) [\alpha/X]) \end{aligned}$$

Then we have to prove that:

$$\begin{aligned} & (W''', (\langle E_{11} [\alpha^{E_1}/X], E_{12} [\alpha^{E_1}/X] \rangle t_1 [\alpha^{E_1}/X] :: \rho(G'_1) [\alpha/X]), (\langle E_{21} [\alpha^{E_2}/X], E_{22} [\alpha^{E_2}/X] \rangle t_2 [\alpha^{E_2}/X] :: \rho(G'_1) [\alpha/X]) \\ & \in \mathcal{J}_{\rho[X \mapsto \alpha]}[G'_1] \end{aligned}$$

Also by the Proposition A.107 we know that:

$$(\varepsilon_i \circ \varepsilon_i^\rho) [\alpha^{E_i}] = (\varepsilon_i [\alpha^{E_i}]) \circ (\varepsilon_i^\rho [\alpha^{E_i}])$$

Note that:

$$(\varepsilon_i \circ \varepsilon_i^\rho) [\alpha^{E_i}] = \langle E_{i1} [\alpha^{E_i}/X], E_{i2} [\alpha^{E_i}/X] \rangle = (\varepsilon_i [\alpha^{E_i}]) \circ (\varepsilon_i^\rho [\alpha^{E_i}])$$

Then we have to prove that:

$$(W''', (\varepsilon_1 [\alpha^{E_1}] \circ \varepsilon_1^\rho [\alpha^{E_1}]) t_1 [\alpha^{E_1}/X] :: G'_1 [\alpha/X], (\varepsilon_2 [\alpha^{E_2}] \circ \varepsilon_2^\rho [\alpha^{E_2}]) t_2 [\alpha^{E_2}/X] :: \rho(G'_1) [\alpha/X])$$

$$\in \mathcal{J}_{\rho[X \mapsto \alpha]} \llbracket G'_1 \rrbracket$$

We know that

$$\begin{aligned} W''' . \Xi_1 \triangleright (\varepsilon_1[\alpha^{E_1}] ; \varepsilon'_1[\alpha^{E_1}]) t_1 &:: G'_1[\alpha/X] \mapsto^{k^*} \\ \Xi_1 \triangleright (\varepsilon_1[\alpha^{E_1}] ; \varepsilon'_1[\alpha^{E_1}]) v_{1f} &:: G'_1[\alpha/X] \mapsto^{k+l} \\ \Xi_1 \triangleright v_1^* & \end{aligned}$$

Note that by the reduction rule of type application, we obtain that:

$$\begin{aligned} W'' . \Xi_i \triangleright (\varepsilon_i \Lambda X . t_i :: \forall X . \rho(G''_1)) [G_i^*] &\longrightarrow \\ W'' . \Xi_i, \alpha := G_i^* \triangleright \varepsilon_{\forall X . \rho(G''_1)}^{E_i/\alpha^{E_i}} (\varepsilon_i[\alpha^{E_i}] t_i[\alpha^{E_i}/X] &:: \rho(G''_1)[\alpha/X] :: \rho(G''_1)[G_i^*/X]) \end{aligned}$$

Note that the evidence ε_i has the form: $\langle \forall X . E''_{i1}, \forall X . E''_{i2} \rangle$, then:

$$\begin{aligned} \varepsilon_{\forall X . \rho(G''_1)}^{E_i/\alpha^{E_i}} (\varepsilon_i[\alpha^{E_i}] t_i[\alpha^{E_i}/X] &:: \rho(G''_1)[\alpha/X] :: \rho(G''_1)[G_i^*/X] = \\ \varepsilon_{\forall X . \rho(G''_1)}^{E_i/\alpha^{E_i}} (\langle E''_{i1}[\alpha^{E_i}/X], E''_{i2}[\alpha^{E_i}/X] \rangle t_i[\alpha^{E_i}/X] &:: \rho(G''_1)[\alpha/X]) \end{aligned}$$

As we know that $(W, v_1, v_2) \in \mathcal{V}_\rho \llbracket \forall X . G''_1 \rrbracket$, then we can instantiate with $\forall W'' \succeq W, G_1^*, G_2^*, R, \varepsilon_1[\alpha^{E_1}] t_1[\alpha^{E_1}/X] :: \rho(G''_1)[\alpha/X], \varepsilon_2[\alpha^{E_2}] t_2[\alpha^{E_2}/X] :: \rho(G''_1)[\alpha/X], \varepsilon_{\forall X . \rho(G''_1)}^{E_1/\alpha^{E_1}}$ and $\varepsilon_{\forall X . \rho(G''_1)}^{E_2/\alpha^{E_2}}$.

Then we know that:

$$(W''', \varepsilon_1[\alpha^{E_1}] t_1[\alpha^{E_1}/X] :: \rho(G''_1)[\alpha/X], \varepsilon_2[\alpha^{E_2}] t_2[\alpha^{E_2}/X] :: \rho(G''_1)[\alpha/X]) \in \mathcal{J}_{\rho[X \mapsto \alpha]} \llbracket G''_1 \rrbracket$$

If the following term reduces to error, then the result follows immediately.

$$W''' . \Xi_1 \triangleright \varepsilon_1[\alpha^{E_1}] t_1[\alpha^{E_1}/X] :: \rho(G''_1)[\alpha/X]$$

If the above is not true, then the following terms reduce to values (v'_{if}) and $\exists W'''' \succeq W'''$ such that $(W'''' , v'_{1f}, v'_{2f}) \in \mathcal{V}_{\rho[X \mapsto \alpha]} \llbracket G''_1 \rrbracket$ and $W'''' . j + k^* + m = W''' . j$.

$$W''' . \Xi_i \triangleright \varepsilon_i[\alpha^{E_i}] t_i[\alpha^{E_i}/X] :: \rho(G''_1)[\alpha/X] \longrightarrow^* W'''' . \Xi_i \triangleright v'_{if}$$

Note that

$$\begin{aligned} W''' . \Xi_1 \triangleright \varepsilon_1[\alpha^{E_1}] t_1[\alpha^{E_1}/X] &:: \rho(G''_1)[\alpha/X] \longrightarrow^{k^*} \\ W'''' . \Xi_1 \triangleright \varepsilon_1[\alpha^{E_1}] v_{1f} &:: \rho(G''_1)[\alpha/X] \longrightarrow^m \end{aligned}$$

$$W''''.\Xi_i \triangleright v'_{1f}$$

By definition of consistency and the evidence we know that $\varepsilon[X] \vdash W''''.\Xi; \Delta, X \vdash G'_1 \sim G'_1$. Then we instantiate the induction hypothesis in the previous result with $G = G'_1$ and $\varepsilon = \varepsilon[X]$. Calling $\rho' = \rho[X \mapsto \alpha]$, then we obtain that:

$$(W''''', \rho'_1(\varepsilon[X])v_{1f} :: \rho'(G'_1), \rho'_2(\varepsilon[X])v_{2f} :: \rho'(G'_1)) \in \mathcal{J}_{\rho'}[[G'_1]]$$

but as $\rho'_1(\varepsilon[X]) = \varepsilon_i^\rho[\alpha^{E_i}]$ which is equivalent to

$$(W''''', (\varepsilon_1^\rho[\alpha^{E_1}])v_{1f} :: \rho(G'_1)[\alpha/X], (\varepsilon_2^\rho[\alpha^{E_2}])v_{2f} :: \rho(G'_1)[\alpha/X]) \in \mathcal{J}_{\rho'}[[G'_1]]$$

Therefore,

$$(\downarrow_k W''''', v_1^*, v_2^*) \in \mathcal{J}_{\rho'}[[G'_1]]$$

where $(\downarrow_k W''''').j + k^* + k + m = W'''''.j$, and the result follows immediately.

Case (Pairs: $G = G_1 \times G_2$). Similar to function case.

Case (A)(Type Names: $G = \alpha$). This means that $\alpha \in \text{dom}(\Xi)$. We know that $(W, \varepsilon_1 u_1 :: \alpha, \varepsilon_2 u_2 :: \alpha) \in \mathcal{V}_\rho[[\alpha]]$ and $\varepsilon_i \vdash W.\Xi_i \vdash G_i \sim \alpha$, then $\varepsilon_i = \langle E_i, \alpha^{E_i} \rangle$. Also we know that $\varepsilon \vdash \Xi; \Delta \vdash \alpha \sim G'$, therefore $\varepsilon = \langle \alpha^{E_1}, E_2^* \rangle$, and $\varepsilon_i^\rho = \langle \alpha^{E_1}, E_2^* \rangle = \varepsilon$, because ε can not have free type variable, so $\varepsilon \vdash \Xi \vdash \alpha \sim G'$. Since $(W, v_1, v_2) \in \mathcal{V}_\rho[[\alpha]]$, we instantiate its definition with $\varepsilon \vdash \Xi \vdash \alpha \sim G'$, Ξ , such that $W \in \mathcal{S}[[\Xi]]$ and G' . Therefore, we know that $(W, \varepsilon v_1 :: G', \varepsilon v_2 :: G')$, and the results follows immediately.

Case (B)(Type Variables: $G = X$). Suppose that $\rho(X) = \alpha$. We know that $\alpha \notin \Xi$, i.e. α may not be in sync, that $(W, \varepsilon_1 u_1 :: \alpha, \varepsilon_2 u_2 :: \alpha) \in \mathcal{V}_\rho[[X]]$ and that $\varepsilon_i \vdash W.\Xi_i \vdash G_i \sim \alpha$, then $\varepsilon_i = \langle E_i, \alpha^{E_i} \rangle$.

Then by construction of evidences, ε must be either $\langle X, X \rangle$ or $\langle ?, ? \rangle$ (any other case will fail when the meet is computed).

- ($\varepsilon = \langle X, X \rangle$). Then $\varepsilon_i^\rho = \langle \rho_i(X), \rho_i(X) \rangle$. But $\rho_i(X)$ is the type that contains the initial precision for α . Therefore $\alpha^{E_i} \sqsubseteq \rho_i(X)$, and by Lemma A.115, $\varepsilon_i \circledast \varepsilon_i^\rho = \varepsilon_i$ and the result holds immediately by Lemma A.97 (notice that if $G' = ?$ then we have to show that they are related to α which is part of the premise).
- ($\varepsilon = \langle ?, ? \rangle$). By Lemma A.115 ($\varepsilon_i^\rho = \langle ?, ? \rangle$), $\varepsilon_i \circledast \langle ?, ? \rangle = \varepsilon_i$ and the result holds immediately by Lemma A.97.

Case (C)(Unknown: $G = ?$). We know that $(W, \varepsilon_1 u_1 :: ?, \varepsilon_2 u_2 :: ?) \in \mathcal{V}_\rho[[?]]$ and $\varepsilon_i \vdash W.\Xi_i \vdash G_i \sim ?$. We are going to proceed by case analysis on ε_i :

(C.i) ($\varepsilon_i = \langle E_i, \alpha^{E_i} \rangle$). Then this means we know that

$$(W, \varepsilon_1 u_1 :: \alpha, \varepsilon_2 u_2 :: \alpha) \in \mathcal{V}_\rho[[\alpha]]$$

and $\varepsilon_i \vdash W.\Xi_i \vdash G_i \sim \alpha$, then $\varepsilon_i = \langle E_i, \alpha^{E_i} \rangle$.

- (a) (Case $\varepsilon = \langle \alpha^{E_3}, E_4 \rangle$). Then as $\langle E_i, \alpha^{E_i} \rangle \Vdash \Xi; \Delta \vdash G_i \sim ?$, then by Lemma A.112 $\langle E_i, \alpha^{E_i} \rangle \Vdash \Xi; \Delta \vdash G_i \sim \alpha$. Also we know that $? \sqsubseteq G$, then $G = ?$, and $\alpha \sqsubseteq G$. Finally, we reduce this case to the Case A if $\alpha \in \Xi$ or Case B if $\alpha \notin \Xi$.

- (b) ($\varepsilon = \langle ?, ? \rangle$). Then $G' = ?$, and does $\varepsilon_i \circ \varepsilon = \varepsilon_i$. Then we have to prove that $(\downarrow_k W, \varepsilon_1 u_1 :: ?, \varepsilon_2 u_2 :: ?) \in \mathcal{V}_\rho[[?]]$, and as $\text{const}(\alpha^{E'_i}) = \alpha$ that is equivalent to prove that $(\downarrow_k W, \varepsilon_1 u_1 :: \alpha, \varepsilon_2 u_2 :: \alpha) \in \mathcal{V}_\rho[[\alpha]]$ which follows by the premise and Lemma A.97.
- (c) ($\varepsilon = \langle ?, \beta^{\beta' \dots ?} \rangle$). Where β cannot transitively point to some unsync variable. Then by definition of the transitivity operator, $\varepsilon_i \circ \varepsilon = \langle E''_i, \beta^{E''_i} \rangle$ (where $\langle E_i, \alpha^{E'_i} \rangle \circ \langle ?, \beta^{\beta' \dots ?} \rangle = \langle E''_i, \beta^{E''_i} \rangle$). Then we have to prove that

$$(\downarrow_k W, \langle E''_1, \beta^{E''_1} \rangle u_1 :: G', \langle E''_2, \beta^{E''_2} \rangle u_2 :: G') \in \mathcal{V}_\rho[[G']]$$

where G' is either $?$ or β . In any case this is equivalent to prove that

$$(\downarrow_k W, \langle E''_1, \beta^{E''_1} \rangle u_1 :: \beta, \langle E''_2, \beta^{E''_2} \rangle u_2 :: \beta) \in \mathcal{V}_\rho[[\beta]]$$

Therefore, we have to prove

$$(\downarrow_{k-1} W, \langle E''_1, E''_1 \rangle u_1 :: G'', \langle E''_2, E''_2 \rangle u_2 :: G'') \in \mathcal{V}_\rho[[G'']]$$

where $G'' = W.\Xi_1(\beta) = W.\Xi_2(\beta)$ (note that β is sync). As $\langle E_i, \alpha^{E'_i} \rangle \circ \langle ?, \beta^{\beta' \dots ?} \rangle = \langle E''_i, \beta^{E''_i} \rangle$, then we can reduce the demonstration to prove that:

$$(\downarrow_{k-1} W, (\langle E_1, \alpha^{E'_1} \rangle \circ \langle ?, \beta^{\beta' \dots ?} \rangle) u_1 :: G'', (\langle E_2, \alpha^{E'_2} \rangle \circ \langle ?, \beta^{\beta' \dots ?} \rangle) u_2 :: G'') \in \mathcal{V}_\rho[[G'']]$$

Thus, we reduce this case to this same case (note that we have base case because the sequence ends in $?$).

Also, we have to prove that $(\forall \Xi', \varepsilon', G^*$ such that $(\downarrow_{k-1} W \in \mathcal{S}[[\Xi']] \wedge \varepsilon' \vdash \Xi' \vdash \beta \sim G^*)$, we get that

$$(\downarrow_{k-1} W, \varepsilon'(\langle E''_1, \beta^{E''_1} \rangle u_1 :: \beta) :: G^*, \varepsilon'(\langle E''_2, \beta^{E''_2} \rangle u_2 :: \beta) :: G^*) \in \mathcal{T}_\rho[[G^*]]$$

or what is the same ($(\langle E''_1, \beta^{E''_1} \rangle \circ \varepsilon')$ fails the result follows immediately)

$$(\downarrow_{k-1-k'} W, (\langle E''_1, \beta^{E''_1} \rangle \circ \varepsilon') u_1 :: G^*, (\langle E''_2, \beta^{E''_2} \rangle \circ \varepsilon') u_2 :: G^*) \in \mathcal{V}_\rho[[G^*]]$$

where $\varepsilon' = \langle \beta^{E_1^*}, E_2^* \rangle$, $\varepsilon'.n = k'$ and $G'' = W'.\Xi_1(\beta) = W'.\Xi_2(\beta)$. By definition of transitivity and Lemma A.115, we know that

$$\langle E''_i, \beta^{E''_i} \rangle \circ \langle \beta^{E_1^*}, E_2^* \rangle = \langle E''_i, E''_i \rangle \circ \langle E_1^*, E_2^* \rangle$$

$$\langle E_i, \alpha^{E'_i} \rangle \circ \langle ?, \beta^{\beta' \dots ?} \rangle = \langle E_{1i}^*, \beta^{E_{2i}^*} \rangle = \langle E''_i, E''_i \rangle$$

Thus $G'' = \beta'$ or $G'' = ?$, in any case we know that $(\downarrow_{k-1} W, \langle E''_1, E''_1 \rangle u_1 :: \beta', \langle E''_2, E''_2 \rangle u_2 :: \beta') \in \mathcal{V}_\rho[[\beta']]$.

We know that $\langle E_1^*, E_2^* \rangle \vdash \Xi' \vdash G'' \sim G^*$. Since $\langle E_1^*, E_2^* \rangle \vdash \Xi \vdash G'' \sim G^*$, $\downarrow_{k-1} W \in \mathcal{S}[[\Xi']]$, $(\downarrow_{k-1} W, \langle E''_1, E''_1 \rangle u_1 :: \beta', \langle E''_2, E''_2 \rangle u_2 :: \beta') \in \mathcal{V}_\rho[[\beta']]$, by the definition of $\mathcal{V}_\rho[[\beta']]$, we know that (since $(\langle E''_1, E''_1 \rangle \circ \varepsilon')$ does not fail then $(\langle E''_1, E''_1 \rangle \circ \langle E_1^*, E_2^* \rangle)$ also does not fail by the transitivity rules and $\langle E_1^*, E_2^* \rangle \vdash \Xi' \vdash \beta' \sim G^*$)

$$(\downarrow_{k-1-k'} W, (\langle E''_1, E''_1 \rangle \circ \langle E_1^*, E_2^* \rangle) u_1 :: G^*, (\langle E''_2, E''_2 \rangle \circ \langle E_1^*, E_2^* \rangle) u_2 :: G^*) \in \mathcal{V}_\rho[[G^*]]$$

The result follows immediately.

(d) ($\varepsilon = \langle ?, \beta^? \rangle$). Then by definition of the transitivity operator, $\varepsilon_i \circ \varepsilon = \langle E_i, \beta^{\alpha^{E'_i}} \rangle$. Then we have to prove that

$$(\downarrow_k W, \langle E_1, \beta^{\alpha^{E'_1}} \rangle_{u_1} :: G', \langle E_2, \beta^{\alpha^{E'_2}} \rangle_{u_2} :: G') \in \mathcal{V}_\rho[[G']]$$

where G' is either $?$ or β . In any case this is equivalent to prove that

$$(\downarrow_k W, \langle E_1, \beta^{\alpha^{E'_1}} \rangle_{u_1} :: \beta, \langle E_2, \beta^{\alpha^{E'_2}} \rangle_{u_2} :: \beta) \in \mathcal{V}_\rho[[\beta]]$$

Therefore, we have to prove that

$(\downarrow_{k-1} W, \langle E_1, \alpha^{E'_1} \rangle_{u_1} :: G'', \langle E_2, \alpha^{E'_2} \rangle_{u_2} :: G'') \in \mathcal{V}_\rho[[G'']]$ where $G'' = W.\Xi_1(\beta) = W.\Xi_2(\beta) = ?$ (note that β is sync). Therefore, we have to prove that $(\downarrow_{k-1} W, \langle E_1, \alpha^{E'_1} \rangle_{u_1} :: \alpha, \langle E_2, \alpha^{E'_2} \rangle_{u_2} :: \alpha) \in \mathcal{V}_\rho[[\alpha]]$ which follows immediately by premise and Lemma A.97.

Also, we have to prove that $(\forall \Xi', \varepsilon', G^*$ such that $(\downarrow_{k-1} W \in \mathcal{S}[[\Xi']] \wedge \varepsilon' \vdash \Xi' \vdash \beta \sim G^*)$, we get that

$$(\downarrow_{k-1} W, \varepsilon'(\langle E_1, \beta^{\alpha^{E'_1}} \rangle_{u_1} :: \beta) :: G^*, \varepsilon'(\langle E_2, \beta^{\alpha^{E'_2}} \rangle_{u_2} :: \beta) :: G^*) \in \mathcal{I}_\rho[[G^*]]$$

or what is the same ($(\langle E_1, \beta^{\alpha^{E'_1}} \rangle \circ \varepsilon')$ fails the result follows immediately)

$$(\downarrow_{k-1-k'} W, (\langle E_1, \beta^{\alpha^{E'_1}} \rangle \circ \varepsilon')_{u_1} :: G^*, (\langle E_2, \beta^{\alpha^{E'_2}} \rangle \circ \varepsilon')_{u_2} :: G^*) \in \mathcal{V}_\rho[[G^*]]$$

where $\varepsilon' = \langle \beta^{E'_1}, E'_2 \rangle$, $\varepsilon'.n = k'$ and $G'' = W'.\Xi_1(\beta) = W'.\Xi_2(\beta) = ?$. By definition of transitivity and Lemma A.115, we know that

$$\langle E_i, \beta^{\alpha^{E'_i}} \rangle \circ \langle \beta^{E'_1}, E'_2 \rangle = \langle E_i, \alpha^{E'_i} \rangle \circ \langle E'_1, E'_2 \rangle$$

We know that $\langle E'_1, E'_2 \rangle \vdash \Xi' \vdash G'' \sim G^*$. Since $\langle E'_1, E'_2 \rangle \vdash \Xi \vdash G'' \sim G^*$, $\downarrow_{k-1} W \in \mathcal{S}[[\Xi]]$, $(\downarrow_{k-1} W, \langle E_1, \alpha^{E'_1} \rangle_{u_1} :: \alpha, \langle E_2, \alpha^{E'_2} \rangle_{u_2} :: \alpha) \in \mathcal{V}_\rho[[\alpha]]$, by the definition of $\mathcal{S}[[\Xi]]\alpha$, we know that (since $(\langle E_1, \alpha^{E'_1} \rangle \circ \varepsilon')$ does not fail then $(\langle E_1, \alpha^{E'_1} \rangle \circ \langle E'_1, E'_2 \rangle)$ also does not fail by the transitivity rules and $\langle E'_1, E'_2 \rangle \vdash \Xi' \vdash \alpha \sim G^*$)

$$(\downarrow_{k-1-k'} W, (\langle E_1, \alpha^{E'_1} \rangle \circ \langle E'_1, E'_2 \rangle)_{u_1} :: G^*, (\langle E_2, \alpha^{E'_2} \rangle \circ \langle E'_1, E'_2 \rangle)_{u_2} :: G^*) \in \mathcal{V}_\rho[[G^*]]$$

The result follows immediately.

(C.ii) ($\varepsilon_i = \langle H_{i1}, H_{i2} \rangle$). Let $G'' = \text{const}(H_{i2})$, and we know that $G'' \in \text{HEADTYPE}$. By unfolding of the logical relation for $?$, we also know that

$$(W, \langle H_{11}, H_{12} \rangle_{u_1} :: G'', \langle H_{21}, H_{22} \rangle_{u_2} :: G'') \in \mathcal{V}_\rho[[G'']]$$

and we have to prove that

$$(\downarrow_k W, (\langle H_{11}, H_{12} \rangle \circ \varepsilon'_1)_{u_1} :: G', (\langle H_{21}, H_{22} \rangle \circ \varepsilon'_2)_{u_2} :: G') \in \mathcal{V}_\rho[[G']]$$

Note that for consistent transitivity to hold, then ε has to take the following forms:

- (a) $\varepsilon = \langle H_3, E_4 \rangle$. Then as $\varepsilon \Vdash \Xi; \Delta \vdash ? \sim G'$, by Lemma A.112, $\varepsilon \Vdash \Xi; \Delta \vdash \text{const}(H_3) \sim G'$, and we proceed just like Case D, where $G \in \text{HEADTYPE}$ ($G = G''$).
- (b) $\varepsilon = \langle ?, ? \rangle$. Then $G' = ?$ and $\langle H_{i1}, H_{i2} \rangle \circ \langle ?, ? \rangle = \langle H_{i1}, H_{i2} \rangle$. The result follows immediately by premise and Lemma A.97.

- (c) $\varepsilon = \langle ?, \alpha^? \rangle$. Then we know that $W.\Xi_i(\alpha) = ?$, and by inspection of the consistent transitivity rules, $\langle H_{i1}, H_{i2} \rangle \circ \langle ?, \alpha^? \rangle = \langle H_{i1}, \alpha^{H_{i2}} \rangle$. Then by definition of the interpretation of G' , which may be $?$ or α , in any case, we have to prove that

$$(\downarrow_k W, \langle H_{11}, \alpha^{H_{12}} \rangle u_1 :: \alpha, \langle H_{21}, \alpha^{H_{22}} \rangle u_2 :: \alpha) \in \mathcal{V}_\rho[\alpha]$$

Therefore, we have to prove that $(\downarrow_{k-1} W, \langle H_{11}, H_{12} \rangle u_1 :: ?, \langle H_{21}, H_{22} \rangle u_2 :: ?) \in \mathcal{V}_\rho[?]$ which follows by premise and Lemma A.97.

Also, we have to prove that $(\forall \Xi', \varepsilon', G^*$ such that $(\downarrow_{k-1} W \in \mathcal{S}[\Xi']) \wedge \varepsilon' \vdash \Xi' \vdash \beta \sim G^*$), we get that

$$(\downarrow_{k-1} W, \varepsilon'(\langle H_{11}, \alpha^{H_{12}} \rangle u_1 :: \alpha) :: G^*, \varepsilon'(\langle H_{21}, \alpha^{H_{22}} \rangle u_2 :: \alpha) :: G^*) \in \mathcal{T}_\rho[G^*]$$

or what is the same $((\langle H_{11}, \alpha^{H_{12}} \rangle \circ \varepsilon')$ fails the result follows immediately)

$$(\downarrow_{k-1-k'} W, (\langle H_{11}, \alpha^{H_{12}} \rangle \circ \varepsilon') u_1 :: G^*, (\langle H_{21}, \alpha^{H_{22}} \rangle \circ \varepsilon') u_2 :: G^*) \in \mathcal{V}_\rho[G^*]$$

where $\varepsilon' = \langle \alpha^{H_1^*}, E_2^* \rangle$, $\varepsilon'.n = k'$. By definition of transitivity and Lemma A.115, we know that

$$\langle H_{i1}, \alpha^{H_{i2}} \rangle \circ \langle \alpha^{H_1^*}, E_2^* \rangle = \langle H_{i1}, H_{i2} \rangle \circ \langle H_1^*, E_2^* \rangle$$

Therefore, we have to prove that

$$(\downarrow_{k-1-k'} W, (\langle H_{11}, H_{12} \rangle \circ \langle H_1^*, E_2^* \rangle) u_1 :: G^*, (\langle H_{21}, H_{22} \rangle \circ \langle H_1^*, E_2^* \rangle) u_2 :: G^*) \in \mathcal{V}_\rho[G^*]$$

We know that $\langle E_1^*, E_2^* \rangle \vdash \Xi' \vdash ? \sim G^*$. Since $\langle E_1^*, E_2^* \rangle \vdash \Xi \vdash ? \sim G^*$, $\downarrow_{k-1} W \in \mathcal{S}[\Xi']$, we follow by this Case(a), but with evidence $\langle H_1^*, E_2^* \rangle$. The result follows immediately.

- (d) $\varepsilon = \langle ?, \alpha^{\beta^{E_4}} \rangle$. Then we know that $W.\Xi_i(\alpha) \in \{\beta, ?\}$ ($W.\Xi_i(\alpha) = G_{123}$) and by inspection of the consistent transitivity rules, $\langle H_{i1}, H_{i2} \rangle \circ \langle ?, \alpha^{\beta^{E_{i4}}} \rangle = \langle H'_{i1}, \alpha^{\beta^{E'_{i4}}} \rangle$, where $\langle H_{i1}, H_{i2} \rangle \circ \langle ?, E_{i4} \rangle = \langle H_{i1}, E'_{i4} \rangle$.

Then by definition of the interpretation of α (after one or two unfolding of $G' = ?$), we have to prove that

$$(\downarrow_{k-1} W, (\langle H'_{11}, \beta^{E'_{14}} \rangle u_1 :: G_{123}), (\langle H'_{21}, \beta^{E'_{24}} \rangle u_2 :: G_{123})) \in \mathcal{T}_\rho[G_{123}]$$

or what is the same

$$\begin{aligned} &(\downarrow_{k-1} W, (\langle H_{11}, H_{12} \rangle \circ \langle ?, \beta^{E_{14}} \rangle) u_1 :: \beta, \\ &(\langle H_{21}, H_{22} \rangle \circ \langle ?, \beta^{E_{24}} \rangle) u_2 :: \beta) \in \mathcal{V}_\rho[\beta] \end{aligned}$$

and then we proceed to the same case one more time (notice that the recursion is finite, until we get to the previous sub case).

Also, we have to prove that $(\forall \Xi', \varepsilon', G^*$ such that $(\downarrow_{k-1} W \in \mathcal{S}[\Xi']) \wedge \varepsilon' \vdash \Xi' \vdash \alpha \sim G^*$), we get that

$$(\downarrow_{k-1} W, \varepsilon'(\langle H'_{11}, \alpha^{\beta^{E'_{14}}} \rangle u_1 :: \alpha) :: G^*, \varepsilon'(\langle H'_{21}, \alpha^{\beta^{E'_{24}}} \rangle u_2 :: \alpha) :: G^*) \in \mathcal{T}_\rho[G^*]$$

or what is the same $((\langle H'_{11}, \alpha^{E'_{14}} \rangle \circ \varepsilon')$ fails the result follows immediately)

$$(\downarrow_{k-1-k'} W, (\langle H'_{11}, \alpha^{\beta^{E'_{14}}} \rangle \circ \varepsilon') u_1 :: G^*, (\langle H'_{21}, \alpha^{\beta^{E'_{24}}} \rangle \circ \varepsilon') u_2 :: G^*) \in \mathcal{V}_\rho[G^*]$$

where $\varepsilon' = \langle \alpha^{E_1^*}, E_2^* \rangle$, $\varepsilon'.n = k'$. By definition of transitivity and Lemma A.115, we know that

$$\langle H'_{i1}, \alpha^{\beta^{E'_{i4}}} \rangle \circ \langle \alpha^{E_1^*}, E_2^* \rangle = \langle H'_{i1}, \beta^{E'_{i2}} \rangle \circ \langle E_1^*, E_2^* \rangle$$

Therefore, we have to prove that

$$(\downarrow_{k-1-k'}W, (\langle H'_{11}, \beta^{E'_{14}} \rangle \circ \langle E_1^*, E_2^* \rangle)u_1 :: G^*, (\langle H_{21}, \beta^{E_{24}} \rangle \circ \langle E_1^*, E_2^* \rangle)u_2 :: G^*) \in \mathcal{V}_\rho[[G^*]]$$

We know that $\langle E_1^*, E_2^* \rangle \vdash \Xi' \vdash G_{123} \sim G^*$. Since $\langle E_1^*, E_2^* \rangle \vdash \Xi \vdash G_{123} \sim G^*$, $\downarrow_{k-1}W \in \mathcal{S}[[\Xi']]$, and $(\downarrow_{k-1}W, (\langle H'_{11}, \beta^{E'_{14}} \rangle)u_1 :: G_{123}), (\langle H'_{21}, \beta^{E'_{24}} \rangle)u_2 :: G_{123}) \in \mathcal{T}_\rho[[G_{123}]]$, by instantiating the definition of $\mathcal{V}_\rho[[\beta]]$, the result follows immediately.

Case (D) (Head Types: $G \in \text{HEADTYPE}$). We know that $(W, \varepsilon_1 u_1 :: \rho(G), \varepsilon_2 u_2 :: \rho(G)) \in \mathcal{V}_\rho[[G]]$ and $\varepsilon_i \vdash W.\Xi_i \vdash G_i \sim G$. Also $\varepsilon_i = \langle H_{i1}, H_{i2} \rangle$, for some H_{i1}, H_{i2} . We proceed by case analysis on G' and ε .

(D.i) ($\varepsilon = \langle H_3, \alpha^{E_4} \rangle$). Then $G' = \alpha$, or $G' = ?$. Notice that as α^{E_4} cannot have free type variables therefore H_3 neither. Then $\varepsilon = \rho_i(\varepsilon)$. As α is sync, then let us call $G'' = W.\Xi_i(\alpha)$. In either case $G' = \alpha$, or $G' = ?$, what we have to prove boils down to

$$(\downarrow_k W, (\varepsilon_1 \circ \langle H_3, \alpha^{E_4} \rangle)u_1 :: \alpha, (\varepsilon_2 \circ \langle H_3, \alpha^{E_4} \rangle)u_2 :: \alpha) \in \mathcal{V}_\rho[[\alpha]]$$

Therefore, we have to prove that

$$(\downarrow_{k-1}W, (\varepsilon_1 \circ \langle H_3, E_4 \rangle)u_1 :: G'', (\varepsilon_2 \circ \langle H_3, E_4 \rangle)u_2 :: G'') \in \mathcal{V}_\rho[[G'']]$$

Then we proceed by case analysis on ε :

- (Case $\varepsilon = \langle H_3, \alpha^{\beta^{E_4}} \rangle$). We know that $\alpha \sqsubseteq G'$ and that $\langle H_3, \alpha^{\beta^{E_4}} \rangle \Vdash \Xi; \Delta \vdash G \sim G'$, then by Lemma A.112, we know that $\langle H_3, \alpha^{\beta^{E_4}} \rangle \vdash \Xi; \Delta \vdash G \sim \alpha$. Also by Lemma A.114, $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash G \sim G''$. As $\beta^{E_4} \sqsubseteq G''$, then G'' can either be $?$ or β .
If $G'' = ?$, then by definition of $\mathcal{V}_\rho[[?]]$, we have to prove that the resulting values belong to $\mathcal{V}_\rho[[\beta]]$. Also as $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash G \sim ?$, by Lemma A.112, $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash G \sim \beta$, and then we proceed just like this case once again (this is process is finite as there are no circular references by construction and it ends up in something different to a type name). If $G'' = \beta$ we use an analogous argument as for $G'' = ?$.
- (Case $\varepsilon = \langle H_3, \alpha^{H_4} \rangle$). Then we have to prove that

$$(\downarrow_{k-1}W, (\varepsilon_1 \circ \langle H_3, H_4 \rangle)u_1 :: G'', (\varepsilon_2 \circ \langle H_3, H_4 \rangle)u_2 :: G'') \in \mathcal{V}_\rho[[G'']]$$

By Lemma A.114, $\langle H_3, H_4 \rangle \vdash \Xi; \Delta \vdash G \sim G''$. Then if $G'' = ?$, we proceed as the case $G \in \text{HEADTYPE}$, $G' = ?$ with $\varepsilon = \langle H_3, H_4 \rangle$ (Case (D.ii)). If $G'' \in \text{HEADTYPE}$, we proceed as the case $G \in \text{HEADTYPE}$, $G' \in \text{HEADTYPE}$ with $\varepsilon = \langle H_3, H_4 \rangle$, where $H_3, H_4 \in \text{HEADTYPE}$ (Case (D.iii)).

Also, we have to prove that $(\forall \Xi', \varepsilon', G^*$ such that $(\downarrow_k W \in \mathcal{S}[[\Xi']] \wedge \varepsilon' \vdash \Xi' \vdash \alpha \sim G^*) \wedge \varepsilon' = \langle \alpha^{E_5}, E_6 \rangle \wedge \varepsilon'.n = k'$, we get that

$$(\downarrow_k W, \varepsilon'((\varepsilon_1 \circ \langle H_3, \alpha^{H_4} \rangle)u_1 :: \alpha) :: G^*, \varepsilon'((\varepsilon_2 \circ \langle H_3, \alpha^{H_4} \rangle)u_2 :: \alpha) :: G^*) \in \mathcal{T}_\rho[[G^*]]$$

or what is the same $((\varepsilon_1 \circ \langle H_3, H_4 \rangle) \circ \langle E_5, E_6 \rangle)$ fails the result follows immediately)

$$(\downarrow_{k-k'}W, (\varepsilon_1 \circ (\langle H_3, H_4 \rangle \circ \langle E_5, E_6 \rangle))u_1 :: G^*, (\varepsilon_2 \circ (\langle H_3, H_4 \rangle \circ \langle E_5, E_6 \rangle))u_2 :: G^*) \in \mathcal{V}_\rho[[G^*]]$$

where $(\langle H_3, H_4 \rangle \circ \langle E_5, E_6 \rangle).n = (k+k')$ We know that $(W, \varepsilon_1 u_1 :: \rho(G), \varepsilon_2 u_2 :: \rho(G)) \in \mathcal{V}_\rho[[G]]$, therefore $(\downarrow_k W, \varepsilon_1 u_1 :: \rho(G), \varepsilon_2 u_2 :: \rho(G)) \in \mathcal{V}_\rho[[G]]$, by Lemma A.97, where now $\varepsilon_1.n = l+k$.

Then we apply the induction hypothesis on $(\downarrow_k W, \varepsilon_1 u_1 :: \rho(G), \varepsilon_2 u_2 :: \rho(G)) \in \mathcal{V}_\rho[[G]]$ and the evidence $(\langle H_3, H_4 \rangle \mathbin{\varepsilon} \langle G_5, G_6 \rangle)$, but where $(\langle H_3, H_4 \rangle \mathbin{\varepsilon} \langle G_5, G_6 \rangle).n = k'$. Therefore the results follows immediately:

$$(\downarrow_{k-k'} W, (\varepsilon_1 \mathbin{\varepsilon} (\langle H_3, H_4 \rangle \mathbin{\varepsilon} \langle G_5, G_6 \rangle)) u_1 :: G^*, (\varepsilon_2 \mathbin{\varepsilon} (\langle H_3, H_4 \rangle \mathbin{\varepsilon} \langle G_5, G_6 \rangle)) u_2 :: G^*) \in \mathcal{V}_\rho[[G^*]]$$

(D.ii) ($G' = ?, \varepsilon = \langle H_3, H_4 \rangle$). We have to prove that

$$(\downarrow_k W, (\varepsilon_1 \mathbin{\varepsilon} \rho_1(\varepsilon)) u_1 :: ?, (\varepsilon_2 \mathbin{\varepsilon} \rho_2(\varepsilon)) u_2 :: ?) \in \mathcal{V}_\rho[[?]]$$

which is equivalent to prove that

$$(\downarrow_k W, (\varepsilon_1 \mathbin{\varepsilon} \rho_1(\varepsilon)) u_1 :: H, (\varepsilon_2 \mathbin{\varepsilon} \rho_2(\varepsilon)) u_2 :: H) \in \mathcal{V}_\rho[[H]]$$

for $H = \text{const}(H_{i_2})$ (and $H \in \text{HEADTYPE}$). But notice that as $\varepsilon \vdash \Xi; \Delta \vdash G \sim ?$, then as $H_4 \sqsubseteq H \sqsubseteq ?$, then by Lemma A.112, $\varepsilon \vdash \Xi; \Delta \vdash G \sim H$, then we proceed just like the case $G \in \text{HEADTYPE}$ and $G' \in \text{HEADTYPE}$ (Case (D.iii)).

(D.iii) ($G' \in \text{HEADTYPE}$). These cases are already analyzed, by structural analysis of types (Case $G = G_1'' \rightarrow G_2''$ and $G' = G_1' \rightarrow G_2'$), (Case $G = \forall X. G_1''$ and $G' = \forall X. G_1'$), (Case $G = \langle G_1'', G_2'' \rangle$ and $G' = \langle G_1', G_2' \rangle$) and (Case $G = B$ and $G' = B$).

□

Lemma A.102 (Ascriptions Preserve Relations) *If $(W, v_1, v_2) \in \mathcal{V}_\rho[[G]]$, $\varepsilon \vdash \Xi; \Delta \vdash G \sim G'$, $W \in \mathcal{S}[[\Xi]]$, and $(W, \rho) \in \mathcal{D}[[\Delta]]$, then $(W, \rho_1(\varepsilon)v_1 :: \rho(G'), \rho_2(\varepsilon)v_2 :: \rho(G')) \in \mathcal{T}_\rho[[G']]$.*

PROOF. Direct by Prop. A.101.

□

Lemma A.103 (Associativity of the evidence)

$$(\varepsilon_1 \mathbin{\varepsilon} \varepsilon_2) \mathbin{\varepsilon} \varepsilon_3 = \varepsilon_1 \mathbin{\varepsilon} (\varepsilon_2 \mathbin{\varepsilon} \varepsilon_3)$$

PROOF. By induction on the structure of evidences.

Case $(\varepsilon_1 = \langle E_{11}, \alpha^{E_{12}} \rangle, \varepsilon_2 = \langle \alpha^{E_{21}}, E_{22} \rangle, \varepsilon_3 = \langle E_{31}, E_{32} \rangle)$. By definition of consistent transitivity, we know that

- $(\varepsilon_1 \mathbin{\varepsilon} \varepsilon_2) \mathbin{\varepsilon} \varepsilon_3 = (\langle E_{11}, E_{12} \rangle \mathbin{\varepsilon} \langle E_{21}, E_{22} \rangle) \mathbin{\varepsilon} \langle E_{31}, E_{32} \rangle$
- $\varepsilon_1 \mathbin{\varepsilon} (\varepsilon_2 \mathbin{\varepsilon} \varepsilon_3) = \langle E_{11}, E_{12} \rangle \mathbin{\varepsilon} (\langle E_{21}, E_{22} \rangle \mathbin{\varepsilon} \langle E_{31}, E_{32} \rangle)$

Then by the induction hypothesis $(\langle E_{11}, E_{12} \rangle \mathbin{\varepsilon} \langle E_{21}, E_{22} \rangle) \mathbin{\varepsilon} \langle E_{31}, E_{32} \rangle = \langle E_{11}, E_{12} \rangle \mathbin{\varepsilon} (\langle E_{21}, E_{22} \rangle \mathbin{\varepsilon} \langle E_{31}, E_{32} \rangle)$, and the result follows immediately.

Case $(\varepsilon_1 = \langle E_{11}, E_{12} \rangle, \varepsilon_2 = \langle E_{21}, \alpha^{E_{22}} \rangle, \varepsilon_3 = \langle \alpha^{E_{31}}, E_{32} \rangle)$. Similar to the previous.

Case $(\varepsilon_1 = \langle \alpha^{E_{11}}, E_{12} \rangle, \varepsilon_2 = \langle E_{21}, E_{22} \rangle, \varepsilon_3 = \langle E_{31}, E_{32} \rangle)$. By definition of consistent transitivity, we know that

- $(\varepsilon_1 \mathbin{\varepsilon} \varepsilon_2) \mathbin{\varepsilon} \varepsilon_3 = \langle \alpha^{E_1}, E_2 \rangle \mathbin{\varepsilon} \langle E_{31}, E_{32} \rangle = \langle \alpha^{E_1}, E_2' \rangle$, where $\langle E_1, E_2 \rangle = (\langle E_{11}, E_{12} \rangle \mathbin{\varepsilon} \langle E_{21}, E_{22} \rangle)$, $\langle E_1', E_2' \rangle = (\langle E_{11}, E_{12} \rangle \mathbin{\varepsilon} \langle E_{21}, E_{22} \rangle) \mathbin{\varepsilon} \langle E_{31}, E_{32} \rangle$.

- $\varepsilon_1 \circ (\varepsilon_2 \circ \varepsilon_3) = \langle \alpha^{E_{11}}, E_{12} \rangle \circ (\langle E_{21}, E_{22} \rangle \circ \langle E_{31}, E_{32} \rangle)$
- Note that by the induction hypothesis $\langle E'_1, E'_2 \rangle = (\langle E_{11}, E_{12} \rangle \circ \langle E_{21}, E_{22} \rangle) \circ \langle E_{31}, E_{32} \rangle = \langle E_{11}, E_{12} \rangle \circ (\langle E_{21}, E_{22} \rangle \circ \langle E_{31}, E_{32} \rangle)$

Then, the result follows immediately because $\langle \alpha^{E_{11}}, E_{12} \rangle \circ (\langle E_{21}, E_{22} \rangle \circ \langle E_{31}, E_{32} \rangle) = \langle \alpha^{E'_1}, E'_2 \rangle$.

Case $(\varepsilon_1 = \langle E_{11}, E_{12} \rangle, \varepsilon_2 = \langle E_{21}, E_{22} \rangle, \varepsilon_3 = \langle E_{31}, \alpha^{E_{32}} \rangle)$. Similar to the previous.

Case $(\varepsilon_1 = \langle ?, ? \rangle, \varepsilon_2 = \langle E_{21}, E_{22} \rangle, \varepsilon_3 = \langle E_{31}, E_{32} \rangle)$. Trivially, by definition of consistent transitivity.

Case $(\varepsilon_1 = \langle E_{11}, E_{12} \rangle, \varepsilon_2 = \langle ?, ? \rangle, \varepsilon_3 = \langle E_{31}, E_{32} \rangle)$. Trivially, by definition of consistent transitivity.

Case $(\varepsilon_1 = \langle E_{11}, E_{12} \rangle, \varepsilon_2 = \langle E_{21}, E_{22} \rangle, \varepsilon_3 = \langle ?, ? \rangle)$. Trivially, by definition of consistent transitivity.

Case $(\varepsilon_1 = \langle E_{11}, E_{12} \rangle, \varepsilon_2 = \langle E_{21}, E_{22} \rangle, \varepsilon_3 = \langle ?, ? \rangle)$. Trivially, by definition of consistent transitivity.

The other cases are pretty similar. □

Lemma A.104 *If $(W, t_1, t_2) \in \mathcal{T}_\rho[[G]]$, then $(\downarrow W, t_1, t_2) \in \mathcal{T}_\rho[[G]]$*

PROOF. By definition of $\mathcal{T}_\rho[[G]]$. □

Proposition A.105 $dom(\varepsilon_1 \circ \varepsilon_2) = dom(\varepsilon_2) \circ dom(\varepsilon_1)$

PROOF. Direct by inspection on the inductive definition of consistent transitivity. □

Proposition A.106 $cod(\varepsilon_1 \circ \varepsilon_2) = cod(\varepsilon_1) \circ cod(\varepsilon_2)$

PROOF. Direct by inspection on the inductive definition of consistent transitivity. □

Proposition A.107 $(\varepsilon_1 \circ \varepsilon_2)[E] = \varepsilon_1[E] \circ \varepsilon_2[E]$.

PROOF. Direct by inspection on the inductive definition of consistent transitivity. □

Lemma A.108 *(Optimality of consistent transitivity).*

If $\varepsilon_3 = \varepsilon_1 \circ \varepsilon_2$ is defined, then $\pi_1(\varepsilon_3) \sqsubseteq \pi_1(\varepsilon_1)$ and $\pi_2(\varepsilon_3) \sqsubseteq \pi_2(\varepsilon_2)$.

PROOF. Direct by inspection on the inductive definition of consistent transitivity. □

Lemma A.109 *If $\varepsilon \vdash \Xi; \Delta \vdash G_1 \sim G_2$, $W \in \mathcal{S}[[\Xi]]$ and $(W, \rho) \in \mathcal{D}[[\Delta]]$ then $\varepsilon_i^\rho \vdash W.\Xi_i; \Delta \vdash \rho(G_1) \sim \rho(G_2)$, where $\varepsilon_i^\rho = \rho_i(\varepsilon)$.*

PROOF. Direct by induction on the structure of the types G_1 and G_2 . □

Lemma A.110 *If $\Xi; \Delta; \Gamma \vdash t : G$, $W \in \mathcal{S}[\Xi]$, $(W, \rho) \in \mathcal{D}[\Delta]$ and $(W, \gamma) \in \mathcal{G}_\rho[\Gamma]$ then $W.\Xi_i \vdash \rho(\gamma_i(t)) : \rho(G)$.*

PROOF. Direct by induction on the structure of the term. □

Lemma A.111 *If*

- $\varepsilon_i \Vdash W.\Xi_i \vdash G_i \sim \rho(G)$, $\varepsilon_1 \equiv \varepsilon_2$
- $\varepsilon \Vdash \Xi; \Delta \vdash G \sim G'$
- $W \in \mathcal{S}[\Xi]$, $(W, \rho) \in \mathcal{D}[\Delta]$
- $\forall \alpha \in \Xi. \alpha^{E_i^*} \in p_2(\varepsilon_i) \Rightarrow E_1^* \equiv E_2^*$

then $\varepsilon_1 \mathbin{\text{;}} \rho_1(\varepsilon) \iff \varepsilon_2 \mathbin{\text{;}} \rho_2(\varepsilon)$.

PROOF. We proceed by induction on the judgment $\varepsilon_i \Vdash W.\Xi_i \vdash G_i \sim G$.

Case $(\varepsilon_i = \langle B_i, B_i \rangle)$. Then the result is trivial as by definition of $\varepsilon_1 \equiv \varepsilon_2$, $B_1 = B_2$, therefore $\varepsilon_1 = \varepsilon_2$. As ε cannot have free type variables (otherwise the result holds immediately), proving that $\varepsilon_1 \mathbin{\text{;}} \varepsilon \iff \varepsilon_1 \mathbin{\text{;}} \varepsilon$ is trivial.

Case $(\varepsilon_i = \langle ?, ? \rangle)$. As the combination with $\langle ?, ? \rangle$ never produce runtime errors, the result follows immediately as both operation never fail.

Case $(\varepsilon_i = \langle E_{1i}, \alpha^{E_{2i}} \rangle)$. We branch on two sub cases:

- Case $\alpha \in \Xi$. Then ε has to have the form $\langle \alpha^{E_3}, E_4 \rangle$, $\langle ?, ? \rangle$ or $\langle ?, \beta^{\dots ?} \rangle$ (otherwise the transitivity operator will always fails in both branches). Also E_4 cannot be a type variable X for instance, because X is consistent with only X or $?$, and in either case the evidence gives you X on both sides of the evidence. And α cannot point to a type variable by construction (e.g, type α^X does not exists). Then ε cannot have free type variables, therefore $\rho_i(\varepsilon) = \varepsilon$, and therefore we have to prove: $\varepsilon_1 \mathbin{\text{;}} \varepsilon \iff \varepsilon_2 \mathbin{\text{;}} \varepsilon$. For cases where $\varepsilon = \langle ?, ? \rangle$ or $\varepsilon = \langle ?, \beta^{\dots ?} \rangle$, then as they never produce runtime errors, the result follows immediately as both operation never fail.

The interesting case is $\varepsilon = \langle \alpha^{E_3}, E_4 \rangle$. By definition of transitivity $\langle E_{1i}, \alpha^{E_{2i}} \rangle \mathbin{\text{;}} \langle \alpha^{E_3}, E_4 \rangle = \langle E_{1i}, E_{2i} \rangle \mathbin{\text{;}} \langle E_3, E_4 \rangle$. By Lemma A.114, $\langle E_{1i}, E_{2i} \rangle \Vdash W.\Xi_i \vdash G_i \sim \Xi(\alpha)$ and $\langle E_3, E_4 \rangle \Vdash W.\Xi_i \vdash \Xi(\alpha) \sim G'$. Also we know by premise that $E_{2i} \equiv E_{2i}$, then by induction hypothesis $\langle E_{11}, E_{21} \rangle \mathbin{\text{;}} \langle E_3, E_4 \rangle \iff \langle E_{12}, E_{22} \rangle \mathbin{\text{;}} \langle E_3, E_4 \rangle$, and the result follows immediately.

- Case $\alpha \notin \Xi$. In this case ε has to have the form $\langle X, X \rangle$ (where $\rho_i(\varepsilon) = \langle \text{lift}_{W.\Xi_i}(\alpha), \text{lift}_{W.\Xi_i}(\alpha) \rangle$), $\langle ?, ? \rangle$ or $\langle ?, \beta^{\dots ?} \rangle$, (otherwise the transitivity always fail in both cases). For cases where $\varepsilon = \langle ?, ? \rangle$ or $\varepsilon = \langle ?, \beta^{\dots ?} \rangle$, by the definition of transitivity, they never produce runtime errors, then the result follows immediately as both operation never fail.

If $\varepsilon = \langle X, X \rangle$, by construction of evidence, $\alpha^{E_{2i}} \sqsubseteq \text{lift}_{W.\Xi_i}(\alpha) \sqsubseteq ?$, then by Lemma A.115, we know that $\varepsilon_i \mathbin{\text{;}} \rho_i(\varepsilon) = \varepsilon_i$, and the result holds.

Case $(\varepsilon_i = \langle \alpha^{E_{i1}}, E_{i2} \rangle)$. Then ε has the form $\langle E_3, E_4 \rangle$, where $\rho_i(\varepsilon) = \langle E_{i3}, E_{i4} \rangle$. By the definition of transitivity we know that:

$$\langle \alpha^{E_{i1}, E_{i2}} \rangle \circ \langle E_{i3}, E_{i4} \rangle \iff \langle E_{i1}, E_{i2} \rangle \circ \langle E_{i3}, E_{i4} \rangle$$

Then by the induction hypothesis with:

$$\begin{aligned} \langle E_{i1}, E_{i2} \rangle \Vdash W.\Xi_i \vdash W.\Xi_i(\alpha) \sim \rho(G) \\ \varepsilon \Vdash \Xi; \Delta \vdash G \sim G' \end{aligned}$$

we know that:

$$\langle E_{11}, E_{22} \rangle \circ \langle E_{13}, E_{14} \rangle \iff \langle E_{21}, E_{22} \rangle \circ \langle E_{23}, E_{24} \rangle$$

Then the result follows immediately.

Case $(\varepsilon_i = \langle E_{11i} \rightarrow E_{12i}, E_{21i} \rightarrow E_{22i} \rangle)$. We analyze cases for ε :

- Case $\varepsilon = \langle ?, ? \rangle$ or $\varepsilon = \langle ?, \beta^{\dots?} \rangle$, then transitivity never fails as explained in previous cases.
- Case $\varepsilon = \langle E_{31} \rightarrow E_{32}, E_{41} \rightarrow E_{42} \rangle$. Then $\rho_i(\varepsilon) = \langle E_{31i} \rightarrow E_{32i}, E_{41i} \rightarrow E_{42i} \rangle$. By definition of interior and meet, the definition of transitivity for functions, can be rewritten like this:

$$\frac{\langle E_{41i}, E_{31i} \rangle \circ \langle E_{21i}, E_{11i} \rangle = \langle E_{i3}, E_{i1} \rangle \quad \langle E_{12i}, E_{22i} \rangle \circ \langle E_{32i}, E_{42i} \rangle = \langle E_{i2}, E_{i4} \rangle}{\langle E_{11i} \rightarrow E_{12i}, E_{21i} \rightarrow E_{22i} \rangle \circ \langle E_{31i} \rightarrow E_{32i}, E_{41i} \rightarrow E_{42i} \rangle = \langle E_{i1} \rightarrow E_{i2}, E_{i3} \rightarrow E_{i4} \rangle}$$

Also notice as the definition of interior is symmetrical (as consistency is symmetric), $\langle E_{41i}, E_{31i} \rangle \circ \langle E_{21i}, E_{11i} \rangle = \langle E_{i3}, E_{i1} \rangle$ can be computed as $\langle E_{11i}, E_{21i} \rangle \circ \langle E_{31i}, E_{41i} \rangle = \langle E_{i1}, E_{i3} \rangle$. Also $\varepsilon_1 \equiv \varepsilon_2$ implies that $dom(\varepsilon_1) \equiv dom(\varepsilon_2)$ and $cod(\varepsilon_1) \equiv cod(\varepsilon_2)$. And that $dom(\varepsilon) \Vdash \Xi; \Delta \vdash dom(G') \sim dom(G)$ is equivalent to:

$$\langle \pi_2(dom(\varepsilon)), \pi_1(dom(\varepsilon)) \rangle \Vdash \Xi; \Delta \vdash dom(G) \sim dom(G')$$

where $cod(\varepsilon) \Vdash \Xi; \Delta \vdash cod(G) \sim cod(G')$. The result holds by applying induction hypothesis on:

$$\begin{aligned} \langle E_{11i}, E_{21i} \rangle \Vdash \Xi; \Delta \vdash dom(G_i) \sim dom(\rho(G)) \\ \langle \pi_2(dom(\varepsilon)), \pi_1(dom(\varepsilon)) \rangle \Vdash \Xi; \Delta \vdash dom(G) \sim dom(G') \end{aligned}$$

and

$$\begin{aligned} \langle E_{12i}, E_{22i} \rangle \Vdash \Xi; \Delta \vdash cod(G_i) \sim cod(\rho(G)) \\ cod(\varepsilon) \Vdash \Xi; \Delta \vdash cod(G) \sim cod(G') \end{aligned}$$

- Case $\varepsilon = \langle E_{31} \rightarrow E_{32}, \alpha^{E_{41} \rightarrow E_{42}} \rangle$. Then $\rho_i(\varepsilon) = \langle E_{31i} \rightarrow E_{32i}, \alpha^{E_{41i} \rightarrow E_{42i}} \rangle$. We use a similar argument to the previous item noticing that

$$\frac{\frac{\langle E_{41i}, E_{31i} \rangle \circ \langle E_{21i}, E_{11i} \rangle = \langle E_{i3}, E_{i1} \rangle \quad \langle E_{12i}, E_{22i} \rangle \circ \langle E_{32i}, E_{42i} \rangle = \langle E_{i2}, E_{i4} \rangle}{\langle E_{11i} \rightarrow E_{12i}, E_{21i} \rightarrow E_{22i} \rangle \circ \langle E_{31i} \rightarrow E_{32i}, E_{41i} \rightarrow E_{42i} \rangle = \langle E_{i1} \rightarrow E_{i2}, E_{i3} \rightarrow E_{i4} \rangle}}{\langle E_{11i} \rightarrow E_{12i}, E_{21i} \rightarrow E_{22i} \rangle \circ \langle E_{31} \rightarrow E_{32}, \alpha^{E_{41} \rightarrow E_{42}} \rangle = \langle E_{i1} \rightarrow E_{i2}, \alpha^{E_{i3} \rightarrow E_{i4}} \rangle}$$

and that if $G' = \alpha$ by Lemma A.114

$$\frac{\langle E_{31} \rightarrow E_{32}, E_{41} \rightarrow E_{42} \rangle \Vdash \Xi; \Delta \vdash G \sim \Xi(\alpha)}{\langle E_{31} \rightarrow E_{32}, \alpha^{E_{41} \rightarrow E_{42}} \rangle \Vdash \Xi; \Delta \vdash G \sim \alpha}$$

and if $G' = ?$ by Lemma A.114

$$\frac{\langle E_{31} \rightarrow E_{32}, E_{41} \rightarrow E_{42} \rangle \Vdash \Xi; \Delta \vdash G \sim ?}{\langle E_{31} \rightarrow E_{32}, \alpha^{E_{41} \rightarrow E_{42}} \rangle \Vdash \Xi; \Delta \vdash G \sim ?}$$

Case $(\varepsilon_i = \langle \forall X.E_{1i}, \forall X.E_{2i} \rangle)$.

We proceed similar to the function case using induction hypothesis on the subtypes.

Case $(\varepsilon_i = \langle E_{1i} \times E_{2i}, E_{3i} \times E_{4i} \rangle)$.

We proceed similar to the function case using induction hypothesis on the subtypes. \square

Lemma A.112 *If $\langle E_1, E_2 \rangle \vdash \Xi; \Delta \vdash G_1 \sim G_2$, then*

1. $\forall G_3, \text{unlift}(E_2) \sqsubseteq G_3 \sqsubseteq G_2, \langle E_1, E_2 \rangle \vdash \Xi; \Delta \vdash G_1 \sim G_3$, and
2. $\forall G_3, \text{unlift}(E_1) \sqsubseteq G_3 \sqsubseteq G_1, \langle E_1, E_2 \rangle \vdash \Xi; \Delta \vdash G_3 \sim G_2$

PROOF. By definition of evidence and interior noticing that always $E_i \sqsubseteq G_i$. \square

Lemma A.113 *If $\langle \alpha^{E_1}, E_2 \rangle \vdash \Xi; \Delta \vdash \alpha \sim G$, then $\langle E_1, E_2 \rangle \vdash \Xi; \Delta \vdash \Xi(\alpha) \sim G$.*

PROOF. Direct by definition of interior and evidence. \square

Lemma A.114 *If $\langle E_1, \alpha^{E_2} \rangle \vdash \Xi; \Delta \vdash G \sim \alpha$, then $\langle E_1, E_2 \rangle \vdash \Xi; \Delta \vdash G \sim \Xi(\alpha)$.*

PROOF. Direct by definition of interior and evidence. \square

Lemma A.115 *If $E_2 \sqsubseteq E_3$ then $\langle E_1, E_2 \rangle \wp \langle E_3, E_3 \rangle = \langle E_1, E_2 \rangle$.*

PROOF. We proceed by induction on $\langle E_1, E_2 \rangle$. If $\langle E_3, E_3 \rangle = \langle ?, ? \rangle$ by definition of transitivity the result holds immediately so we do not consider this case in the following.

Case $(\langle E_1, E_2 \rangle = \langle ?, ? \rangle)$. Then we know that $E_3 = ?$, and the result follows immediately.

Case $(\langle E_1, E_2 \rangle = \langle E_1, \alpha^{E'_2} \rangle)$. Then $\langle E_3, E_3 \rangle = \langle \alpha^{E'_3}, \alpha^{E'_3} \rangle$. Then $\langle E_1, \alpha^{E'_2} \rangle \wp \langle \alpha^{E'_3}, \alpha^{E'_3} \rangle$ boils down to $\langle E_1, E'_2 \rangle \wp \langle E'_3, E'_3 \rangle$, if $E'_2 = \beta^{E''_2}$, then E'_3 has to be $\beta^{E''_3}$ and we repeat this process. Let us assume that $E'_2 \notin \text{SITYPENAME}$, then by definition of meet $E'_3 \notin \text{SITYPENAME}$. By definition of precision if $\alpha^{E'_2} \sqsubseteq \alpha^{E'_3}$, then $E'_2 \sqsubseteq E'_3$. Then by induction hypothesis $\langle E_1, E'_2 \rangle \wp \langle E'_3, E'_3 \rangle = \langle E_1, E'_2 \rangle$, then $\langle E_1, \alpha^{E'_2} \rangle \wp \langle \alpha^{E'_3}, \alpha^{E'_3} \rangle = \langle E_1, \alpha^{E'_2} \rangle$ and the result holds.

Case $(\langle E_1, E_2 \rangle = \langle \alpha^{E'_1}, E_2 \rangle)$. Then $\langle \alpha^{E'_1}, E_2 \rangle \wp \langle E_3, E_3 \rangle$ boils down to $\langle E'_1, E_2 \rangle \wp \langle E_3, E_3 \rangle$. We know that $E_2 \sqsubseteq E_3$. Then by induction hypothesis $\langle E'_1, E_2 \rangle \wp \langle E_3, E_3 \rangle = \langle E_1, E'_2 \rangle$, then $\langle \alpha^{E'_1}, E_2 \rangle \wp \langle E_3, E_3 \rangle = \langle \alpha^{E'_1}, E_2 \rangle$ and the result holds.

Case $(\langle E_1, E_2 \rangle = \langle B, B \rangle)$. Then by definition of precision E_3 is either ? (case we wont analyze) or B . But $\langle B, B \rangle \wp \langle B, B \rangle = \langle B, B \rangle$ and the result holds.

Case $(\langle E_1, E_2 \rangle = \langle E_{11} \rightarrow E_{12}, E_{21} \rightarrow E_{22} \rangle)$. Then E_3 has to have the form $E_{31} \rightarrow E_{32}$. By definition of precision, if $E_{21} \rightarrow E_{22} \sqsubseteq E_{31} \rightarrow E_{32}$ then $E_{21} \sqsubseteq E_{31}$ and $E_{22} \sqsubseteq E_{32}$. As $\langle E_{31}, E_{31} \rangle \wp \langle E_{21}, E_{11} \rangle = (\langle E_{11}, E_{21} \rangle \wp \langle E_{31}, E_{31} \rangle)^{-1}$. By induction hypothesis $\langle E_{11}, E_{21} \rangle \wp \langle E_{31}, E_{31} \rangle = \langle E_{11}, E_{21} \rangle$ and $\langle E_{12}, E_{22} \rangle \wp \langle E_{32}, E_{32} \rangle = \langle E_{12}, E_{22} \rangle$. Therefore $\langle E_{11} \rightarrow E_{12}, E_{21} \rightarrow E_{22} \rangle \wp \langle E_{31} \rightarrow E_{32}, E_{31} \rightarrow E_{32} \rangle = \langle E_{11} \rightarrow E_{12}, E_{21} \rightarrow E_{22} \rangle$ and the result holds.

Case $\langle\langle E_1, E_2 \rangle\rangle = \langle\langle \forall X.E_{11}, \forall X.E_{21} \rangle\rangle$ or $\langle\langle E_1, E_2 \rangle\rangle = \langle\langle E_{11} \times E_{12}, E_{21} \times E_{22} \rangle\rangle$. Analogous to function case.

□

A.6.3 Contextual Equivalence

In this section we show that the logical relation is sound with respect to contextual approximation (and therefore contextual equivalence). Figure A.5 presents the syntax and static semantics of contexts.

Definition A.116 (Contextual Approximation and Equivalence)

$$\begin{aligned} \Xi; \Delta; \Gamma \vdash t_1 \preceq^{ctx} t_2 : G &\triangleq \Xi; \Delta; \Gamma \vdash t_1 : G \wedge \Xi; \Delta; \Gamma \vdash t_2 : G \wedge \forall C, \Xi', G'. \\ &\vdash C : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \cdot; \vdash G') \Rightarrow ((\Xi' \triangleright t_1 \Downarrow \Longrightarrow \Xi' \triangleright t_2 \Downarrow) \wedge \\ &(\exists \Xi_1. \Xi' \triangleright C[t_1] \mapsto^* \Xi_1 \triangleright \mathbf{error} \Rightarrow \exists \Xi_2. \Xi' \triangleright C[t_2] \mapsto^* \Xi_2 \triangleright \mathbf{error})) \\ \Xi; \Delta; \Gamma \vdash t_1 \approx^{ctx} t_2 : G &\triangleq \Xi; \Delta; \Gamma \vdash t_1 \preceq^{ctx} t_2 : G \wedge \Xi; \Delta; \Gamma \vdash t_2 \preceq^{ctx} t_1 : G \end{aligned}$$

Theorem A.117 (Soundness w.r.t. Contextual Approximation) *If $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : G$ then $\Xi; \Delta; \Gamma \vdash t_1 \preceq^{ctx} t_2 : G$.*

PROOF. The proof follows the usual route of going through congruence and adequacy.

□

$$\begin{aligned}
C &::= [\cdot] \mid \varepsilon C_u \mid G \mid \langle C, t \rangle \mid \langle t, C \rangle \mid C \ t \mid t \ C \mid \varepsilon C \mid op(\bar{t}, C, \bar{t}) \mid C \ [G] \mid \pi_i(C) \quad (\text{GSF}\varepsilon \text{ Contexts}) \\
C_u &::= \lambda x : G.C \mid \Lambda X.C \mid \langle C_u, u \rangle \mid \langle u, C_u \rangle \\
C_s &::= C \mid C_u
\end{aligned}$$

$\boxed{\vdash C : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G')}$ **Well-typed contexts**

$$\begin{aligned}
(\text{Cid}) &\frac{\Xi \subseteq \Xi' \quad \Delta \subseteq \Delta' \quad \Gamma \subseteq \Gamma' \quad \Xi; \Delta \vdash \Gamma \quad \Xi'; \Delta' \vdash \Gamma'}{\vdash [\cdot] : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G)} \\
(\text{C}\lambda) &\frac{\vdash C : (\Xi; \Delta; \Gamma, x : G_1 \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma', x : G_1 \vdash G_2)}{\vdash \lambda x : G_1.C : (\Xi; \Delta; \Gamma, x : G_1 \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G_1 \rightarrow G_2)} \\
(\text{C}\Lambda) &\frac{\vdash C : (\Xi; \Delta, X; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta', X; \Gamma' \vdash G') \quad \Xi; \Delta \vdash \Gamma \quad \Xi'; \Delta' \vdash \Gamma'}{\vdash \Lambda X.C : (\Xi; \Delta, X; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash \forall X.G')} \\
(\text{CpairL}) &\frac{\vdash C : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G_1) \quad \Xi'; \Delta'; \Gamma' \vdash t : G_2}{\vdash \langle C, t \rangle : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G_1 \times G_2)} \\
(\text{CpairR}) &\frac{\Xi'; \Delta'; \Gamma' \vdash t : G_1 \quad \vdash C : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G_2)}{\vdash \langle t, C \rangle : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G_1 \times G_2)} \\
(\text{Casc}) &\frac{\vdash C_s : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G') \quad \varepsilon \Vdash \Xi; \Delta \vdash G' \sim G''}{\vdash \varepsilon C_s :: G'' : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G'')} \\
(\text{Cop}) &\frac{\Xi'; \Delta'; \Gamma' \vdash \bar{t}_1 : \overline{G_1} \quad \vdash C : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G_3) \quad \Xi'; \Delta'; \Gamma' \vdash \bar{t}_2 : \overline{G_2} \quad ty(op) = (\overline{G_1}, G_3, \overline{G_2}) \rightarrow G''}{\vdash op(\bar{t}_1, C, \bar{t}_2) : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G'')} \\
(\text{CappL}) &\frac{\vdash C : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G_1 \rightarrow G_2) \quad \Xi'; \Delta'; \Gamma' \vdash t : G_1}{\vdash C \ t : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G_2)} \\
(\text{CappR}) &\frac{\Xi'; \Delta'; \Gamma' \vdash t : G_1 \rightarrow G_2 \quad \vdash C : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G_1)}{\vdash t \ C : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G_2)} \\
(\text{CappG}) &\frac{\vdash C : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash \forall X.G') \quad \Xi'; \Delta' \vdash G''}{\vdash C \ [G''] : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G'[G''/X])} \\
(\text{Cpairi}) &\frac{\vdash C : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G_1 \times G_2)}{\vdash \pi_i(C) : (\Xi; \Delta; \Gamma \vdash G) \rightsquigarrow (\Xi'; \Delta'; \Gamma' \vdash G_i)}
\end{aligned}$$

Figure A.5: GSF ε : Syntax and Static Semantics - Contexts

A.7 Parametricity vs. the DGG in GSF

In this section, we present the proofs of the auxiliary Lemmas need to show that the definition of parametricity for GSF is incompatible with the DGG.

Lemma A.118 *Let $\vdash (\Lambda X.\lambda x : ?.t) \rightsquigarrow v_a : \forall X. ? \rightarrow X$ and $\vdash v \rightsquigarrow v_b : ?$. Let G_1 and G_2 , such that $\text{const}(G_1) \neq \text{const}(G_2)$. If $\cdot \triangleright v_a [G_i] \mapsto \alpha := G_i \triangleright \varepsilon_i v_i :: ? \rightarrow G_i$ and $\varepsilon_i \Vdash ? \rightarrow \alpha \sim ? \rightarrow G_i$, then $\forall W \in \mathcal{S}[\cdot], \forall R \in \text{REL}_{W,j}[G_1, G_2], (W \boxtimes (\alpha, G_1, G_2, R), \text{dom}(\varepsilon_1)v_b :: ?, \text{dom}(\varepsilon_2)v_b :: ?) \in \mathcal{I}_{X \mapsto \alpha}[\cdot]$.*

PROOF. Notice that v_a has to be of the form $(\varepsilon'(\Lambda X.\varepsilon''(\lambda x : ?.t') :: ? \rightarrow X) :: \forall X. ? \rightarrow X)$, where $\varepsilon' = \langle \forall X. ? \rightarrow X, \forall X. ? \rightarrow X \rangle$ and $\varepsilon'' = \langle ? \rightarrow X, ? \rightarrow X \rangle$. Then $\cdot \triangleright v_a [G_i] \mapsto \langle ? \rightarrow \hat{\alpha}_i, ? \rightarrow E_i \rangle t'$ for some t' , where $\hat{\alpha}_i = \text{lift}_{\alpha \mapsto G_i}(\alpha)$ and $E_i = \text{lift}(G_i)$. We know that $\cdot; \cdot \vdash v_b : ?$ then as $X \notin \text{FTV}(v)$, $\cdot; X; \cdot \vdash v_b : ?$, therefore by the fundamental property (Thm 3.28), $\cdot; X; \cdot \vdash v_b \preceq v_b : ?$, therefore as $W \in \mathcal{S}[\cdot]$, we can pick $W' = W \boxtimes (\alpha, G_1, G_2, R) \in \mathcal{S}[\cdot]$, and $(W', X \mapsto \alpha) \in \mathcal{D}[X]$ and thus conclude that $(W', v_b, v_b) \in \mathcal{I}_{X \mapsto \alpha}[\cdot]$. Now notice that $\text{dom}(\varepsilon_i) = \langle ?, ? \rangle$, but $\varepsilon \circ \langle ?, ? \rangle = \varepsilon$ for any evidence ε , therefore $\alpha := G_i \triangleright \text{dom}(\varepsilon_i)v_b :: ? \mapsto \alpha := G_i \triangleright v_b$, then we have to prove that $(\downarrow W', v_b, v_b) \in \mathcal{I}_{X \mapsto \alpha}[\cdot]$ which follows directly from the weakening lemma. \square

Lemma A.119 *For any $\vdash v : ?$ and $\vdash G$, we have $(\Lambda X.\lambda x : ?.x :: X) [G] v \Downarrow \mathbf{error}$.*

PROOF. Let $\text{id}_? \triangleq \Lambda X.\lambda x : ?.x :: X$, $\vdash \text{id}_? \rightsquigarrow v_a : \forall X. ? \rightarrow X$, and v s.t. $\vdash v \rightsquigarrow v_b : ?$.

By the fundamental property (Th. 3.28), $\vdash v_a \preceq v_a : \forall X. ? \rightarrow X$ so for any $W_0 \in \mathcal{S}[\cdot]$, $(W_0, v_a, v_a) \in \mathcal{I}_\emptyset[\forall X. ? \rightarrow X]$. Because v_a is a value, $(W_0, v_a, v_a) \in \mathcal{V}_\emptyset[\forall X. ? \rightarrow X]$. By reduction, $\cdot \triangleright v_a [G_i] \mapsto^* \Xi'_i \triangleright \varepsilon'_i v_i :: ? \rightarrow G_i$ for some $\varepsilon'_i, \varepsilon_i$ and $\varepsilon_{i\alpha}$, where $\Xi'_i = \{\alpha := G_i\}$ and $v_i = \varepsilon_i(\lambda x : ?.(\varepsilon_{i\alpha} x :: \alpha)) :: ? \rightarrow \alpha$. We can instantiate the definition of $\mathcal{V}_\emptyset[\forall X. ? \rightarrow X]$ with $W_0, G_1 = G$ and G_2 structurally different (and different from $?$), some $R \in \text{REL}_{W_0,j}[G_1, G_2], v_1, v_2, \varepsilon'_1$ and ε'_2 , then we have that $(W_1, v_1, v_2) \in \mathcal{I}_{X \mapsto \alpha}[\cdot \rightarrow X]$, where $W_1 = (\downarrow(W_0 \boxtimes (\alpha, G_1, G_2, R)))$. As v_1 and v_2 are values, $(W_1, v_1, v_2) \in \mathcal{V}_{X \mapsto \alpha}[\cdot \rightarrow X]$. Also, by associativity of consistent transitivity, the reduction of $\Xi'_i \triangleright (\varepsilon'_i v_i :: ? \rightarrow G_i) v_?$ is equivalent to that of $\Xi'_i \triangleright \text{cod}(\varepsilon'_i)(v_i (\text{dom}(\varepsilon'_i)v_? :: ?)) :: G_i$.

By the fundamental property (Th. 3.28) we know that $\vdash v_b \preceq v_b : ?$; we can instantiate this definition with W_0 , and we have that $(W_0, v_b, v_b) \in \mathcal{V}_\emptyset[\cdot]$. By Lemma 3.34, $(W_1, \text{dom}(\varepsilon'_1)v_? :: ?, \text{dom}(\varepsilon'_2)v_? :: ?) \in \mathcal{I}_{X \mapsto \alpha}[\cdot]$. If $\text{dom}(\varepsilon'_1)v_? :: ?$ reduces to \mathbf{error} then the result follows immediately. Otherwise, $\Xi'_i \triangleright \text{dom}(\varepsilon'_1)v_? :: ? \mapsto^* \Xi'_i \triangleright v''_i$, and $(W_2, v''_1, v''_2) \in \mathcal{V}_{X \mapsto \alpha}[\cdot]$, where $W_2 = \downarrow W_1$, and some v''_1 and v''_2 . We can instantiate the definition of $\mathcal{V}_{X \mapsto \alpha}[\cdot \rightarrow X]$ with W_2, v''_1 and v''_2 , obtaining that $(W_2, v_1 v''_1, v_2 v''_2) \in \mathcal{I}_{X \mapsto \alpha}[X]$. We then proceed by contradiction. Suppose that $\Xi'_i \triangleright v_i v''_i \mapsto^* \Xi'_i \triangleright v'_i$ (for a big-enough step index). If $v''_i = \varepsilon''_{iv} u :: ?$, then by evaluation $v'_i = \varepsilon'_{iv} u :: \alpha$, for some ε'_{iv} . But by definition of $\mathcal{V}_{X \mapsto \alpha}[X]$, it must be the case that for some $W_3 \succeq W_2, (W_3, \varepsilon'_{1v} u :: G_1, \varepsilon'_{2v} u :: G_2) \in R$, which is impossible because u cannot be ascribed to structurally different types G_1 and G_2 . Therefore $v_1 v''_1$ cannot reduce to a value, and hence the term $v_a [G] v_b$ cannot reduce to a value either. Because v_a is non-diverging, its application must produce \mathbf{error} . \square

A.8 A Cheap Theorem in GSF

This section shows the proof of the cheap theorem presented in the paper and some auxiliary results.

Definition A.120 Let $\mathcal{X}(t, \alpha)$ a predicate that holds if and only if in each evidence of term t , if α is present, then it appears on both sides of the evidence and in the same structural position. This predicate is defined inductively as follows:

$$\frac{\forall \varepsilon \in t, \mathcal{X}(\varepsilon, \alpha)}{\mathcal{X}(t, \alpha)}$$

where

$$\frac{}{\mathcal{X}(\langle \alpha^E, \alpha^E \rangle, \alpha)} \quad \frac{\alpha \notin \text{FTN}(E_1) \cup \text{FTN}(E_2)}{\mathcal{X}(\langle E_1, E_2 \rangle, \alpha)} \quad \frac{\mathcal{X}(\langle E_1, E_3 \rangle, \alpha) \quad \mathcal{X}(\langle E_2, E_4 \rangle, \alpha)}{\mathcal{X}(\langle E_1 \rightarrow E_2, E_3 \rightarrow E_4 \rangle, \alpha)}$$

$$\frac{\mathcal{X}(\langle E_1, E_3 \rangle, \alpha) \quad \mathcal{X}(\langle E_2, E_4 \rangle, \alpha)}{\mathcal{X}(\langle E_1 \times E_2, E_3 \times E_4 \rangle, \alpha)} \quad \frac{\mathcal{X}(\langle E_1, E_2 \rangle, \alpha)}{\mathcal{X}(\langle \forall X. E_1, \forall X. E_2 \rangle, \alpha)}$$

Corollary A.121 Let t and v be static terms such that $\vdash t : \forall X. T$, $\vdash v : T'$, and $t [T'] v \Downarrow v'$.

1. If $\forall X. T \sqsubseteq \forall X. X \rightarrow ?$ then $(t :: \forall X. X \rightarrow ?) [T'] v \Downarrow v''$, and $v' \leq v''$.
2. If $\forall X. T \sqsubseteq \forall X. ? \rightarrow X$ then $(t :: \forall X. ? \rightarrow X) [T'] v \Downarrow v''$, and $v' \leq v''$.

PROOF. Direct by Lemmas 3.21 and 3.24. □

Lemma A.122 $\forall W \in \mathcal{S}[\Xi], \rho, \gamma. ((W, \rho) \in \mathcal{D}[\Delta] \wedge (W, \gamma) \in \mathcal{G}_\rho[\Gamma])$, such that $\forall v \in \text{cod}(\gamma_i)$, $\mathcal{X}(v, \alpha)$. If $\mathcal{X}(\rho(\gamma_i(t_i)), \alpha)$, then $\Xi \triangleright \rho(\gamma_i(t_i)) \mapsto \Xi' \triangleright t'_i$ and $\mathcal{X}(t', \alpha)$

PROOF. By induction on the structure of t_i . The proof is direct by looking at the inductive definition of construction of evidences (interior), noticing that $\forall G, \mathcal{G}(\cdot, \cdot)X, G = \mathcal{G}(\cdot, \cdot)G, X = \langle X, X \rangle$. Then by inspection of consistent transitivity we know that, for any evidence of a value $\langle E_1, E_2 \rangle$

$$\langle E_1, E_2 \rangle ; \langle \alpha^E, \alpha^E \rangle = \langle E'_1, \alpha^{E'} \rangle \wedge E'_1 \neq \alpha^* \iff E_2 = \alpha^{E''} \wedge E_1 \neq \alpha^*$$

but if that is the case $\neg(\mathcal{X}(\langle E_1, E_2 \rangle, \alpha))$, which contradicts the premise. □

Theorem A.123 Let $v \triangleq \Lambda X. \lambda x : ?. t$ for some t , such that $\vdash v : \forall X. ? \rightarrow X$. Then for any $\vdash v' : G$, we either have $v [G] v' \Downarrow \mathbf{error}$ or $v [G] v' \Uparrow$.

PROOF. Let $\vdash v \rightsquigarrow v_\forall : \forall X. ? \rightarrow X$, $\vdash v' \rightsquigarrow v_? : ?$. Because $\vdash v_\forall : \forall X. ? \rightarrow X$ and $\vdash v_? : ?$, by the fundamental property (Theorem 3.28) we know that

$$(W_0, v_\forall, v_?) \in \mathcal{V}_\emptyset[\forall X. ? \rightarrow X]$$

$$(W_0, v_?, v_?) \in \mathcal{V}_\emptyset[?]$$

Let $v_{\forall} = \varepsilon(\Lambda X.(\lambda x : ?.t)) :: \forall X. ? \rightarrow X$, where $\varepsilon \Vdash \cdot ; \cdot \vdash \forall X. ? \rightarrow X \sim \forall X. ? \rightarrow X$, and therefore $\varepsilon = \langle \forall X. ? \rightarrow X, \forall X. ? \rightarrow X \rangle$.

Note that by the reduction rules we know that

$$\Xi \triangleright v_{\forall} [G] \mapsto^* \Xi'_1 \triangleright \varepsilon_1(\varepsilon_2(\lambda x : ?.t') :: ? \rightarrow \alpha) :: ? \rightarrow G$$

for some t' , where $\varepsilon_1 = \langle ? \rightarrow \alpha^E, ? \rightarrow E \rangle$, $\varepsilon_2 = \langle ? \rightarrow \alpha^E, ? \rightarrow \alpha^E \rangle$, $E = \text{lift}(G)$, $\Xi'_1 = \Xi$, $\alpha = G$.

By definition of $\mathcal{V}_{\emptyset}[\forall X. ? \rightarrow X]$ if we pick $G_1 = G_2 = G$, and some R , then for some W_1 we know that $(W_1, v_1, v_2) \in \mathcal{V}_{X \rightarrow \alpha}[\forall X. ? \rightarrow X]$, where $v_i = \varepsilon_2(\lambda x : ?.t') :: ? \rightarrow \alpha$.

Also, by the reduction rules we know that $\Xi'_i \triangleright (\varepsilon_1 v_i :: ? \rightarrow G) v_{?} \iff \Xi'_i \triangleright \text{cod}(\varepsilon_1)(v_i (\text{dom}(\varepsilon_1) v_{?} :: ?)) :: G$. As $\text{dom}(\varepsilon_1) = \langle ?, ? \rangle$, then $\Xi' \triangleright \text{dom}(\varepsilon_1) v_{?} :: ? \mapsto \Xi' \triangleright v_{?} :: ?$. As $\alpha \notin \text{FTN}(v_{?})$, then $\mathcal{X}(v_{?}, \alpha)$. Also we know that $\mathcal{X}(v_i, \alpha)$. Then by Lemma A.122, if $\Xi' \triangleright t'[v_{?}] \mapsto^* v'$, then $\mathcal{X}(v', \alpha)$, but that is a contradiction because if $(W_4, v', v') \in \mathcal{V}_{\rho}[\alpha]$, then $\neg \mathcal{X}(v', \alpha)$ and the result holds. \square

A.9 Embedding Dynamic Sealing in GSF

In this section, we prove Theorem 4.1, using the simulation relation \approx between λ_{seal} and GSF_ε , defined in Figure 4.3. We also define a direct embedding of λ_{seal} into GSF_ε to make the proof simpler.

$$\begin{aligned}
[x]_\varepsilon &= x \\
[\sigma]_\varepsilon &= su_\varepsilon^\sigma \\
[b]_\varepsilon &= \varepsilon_B(\varepsilon_B b :: B) :: ? \\
[\lambda x.t]_\varepsilon &= \varepsilon_{\rightarrow?}(\varepsilon_{\rightarrow?} \lambda x. [t]_\varepsilon :: ? \rightarrow ?) :: ? \\
[\langle t_1, t_2 \rangle]_\varepsilon &= \varepsilon_{\times?}(\langle [t_1]_\varepsilon, [t_2]_\varepsilon \rangle :: ?) \\
[\pi_i(t)]_\varepsilon &= \pi_i(\varepsilon_{\times?} [t]_\varepsilon :: ? \times ?) \\
[\text{op}(\bar{t})]_\varepsilon &= \text{let } \bar{x} : ? = [\bar{t}] \text{ in } \varepsilon_B \text{op}(\varepsilon_{\bar{B}} \bar{x} :: \bar{B}) :: ? \\
[\nu x.t]_\varepsilon &= \text{let } x = su_\varepsilon \text{ in } [t]_\varepsilon \\
[t_1 t_2]_\varepsilon &= \text{let } x = [t_1]_\varepsilon \text{ in let } y = [t_2]_\varepsilon \text{ in } (\varepsilon_{\rightarrow?} x :: ? \rightarrow ?) y \\
[\{t_1\}_{t_2}]_\varepsilon &= \text{let } x = [t_1]_\varepsilon \text{ in let } y = [t_2]_\varepsilon \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) x \\
[\text{let } \{z\}_{t_1} = t_2 \text{ in } t_3]_\varepsilon &= \text{let } x = [t_1]_\varepsilon \text{ in let } y = [t_2]_\varepsilon \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } [t_3]_\varepsilon
\end{aligned}$$

Figure A.6: Compilation from λ_{seal} to GSF_ε

Definition A.124 We said that μ and Ξ are synchronized, denoted $\mu \equiv \Xi$, if and only if $\sigma \in \mu \iff \sigma := ? \in \Xi$.

Lemma A.125 Let t be a λ_{seal} term. If $\Xi; \Gamma \vdash [t] \rightsquigarrow t_\varepsilon : ?$ then $[t]_\varepsilon = t_\varepsilon$.

PROOF. The proof is straightforward by induction on the syntax of t , and following definitions of $[t]$, $\Xi; \Gamma \vdash [t] \rightsquigarrow t_\varepsilon : ?$ and $[t]_\varepsilon$. \square

Lemma A.126 If $\Xi; \Gamma \vdash [t] \rightsquigarrow t_\varepsilon : ?$, then $\mu; \Xi; \Gamma \vdash t \approx t_\varepsilon : ?$, for some $\mu \equiv \Xi$.

PROOF. By Lemma A.125, we know that $t_\varepsilon = [t]_\varepsilon$. Therefore, we are required to prove that $\mu; \Xi; \Gamma \vdash t \approx [t]_\varepsilon : ?$. We follow by induction on the syntax of t . Since translation preserves typing (Theorem 3.12), we know that $\Xi; \Gamma \vdash [t]_\varepsilon : ?$.

Case (x). Then, we know that

$$[x]_\varepsilon = x$$

We have $t = x$. By premise we know that $\Xi; \Gamma \vdash x : ?$ which implies that $x : ? \in \Gamma$ and $\Xi; \vdash \Gamma$. Therefore, $\mu; \Xi; \Gamma \vdash t \approx [t]_\varepsilon : ?$ by Rule (Rx) and the result follows immediately.

Case (b). Then, we know that

$$[b]_\varepsilon = \varepsilon_B(\varepsilon_B b :: B) :: ?$$

We have $t = b$. Then, we have to prove that $\mu; \Xi; \Gamma \vdash b \approx \varepsilon_B(\varepsilon_B b :: B) :: ? : ?$. We know by the Rule (Rb) that $\mu; \Xi; \Gamma \vdash b \approx \varepsilon_B b :: ? : ?$. Therefore, by the Rule (Ru) the result follows immediately.

Case $(\lambda x.t')$. Then, we know that

$$[\lambda x.t']_\varepsilon = \varepsilon_{\rightarrow?}(\varepsilon_{\rightarrow?}\lambda x.[t']_\varepsilon :: ? \rightarrow ?) :: ?$$

We have $t = \lambda x.t'$. Then, we have to prove that $\mu; \Xi; \Gamma \vdash \lambda x.t' \approx \varepsilon_{\rightarrow?}(\varepsilon_{\rightarrow?}\lambda x.[t']_\varepsilon :: ? \rightarrow ?) :: ? : ?$. Since $\Xi; \Gamma \vdash [t]_\varepsilon : ?$ and by Lemma A.137, we know that $\Xi; \Gamma, x : ? \vdash [t']_\varepsilon : ?$, thus by the induction hypothesis $\mu; \Xi; \Gamma, x : ? \vdash t' \approx [t']_\varepsilon : ?$. Therefore, by the Rule (R λ) that $\mu; \Xi; \Gamma \vdash \lambda x.t' \approx \varepsilon_{\rightarrow?}\lambda x.[t']_\varepsilon :: ? : ?$. Therefore, by the Rule (Ru) the result follows immediately.

Case (σ) . Then, we know that

$$[\sigma]_\varepsilon = su_\varepsilon^\sigma$$

We have $t = \sigma$. Then, we have to prove that $\mu; \Xi; \Gamma \vdash \sigma \approx su_\varepsilon^\sigma : ?$. By premise we know that $\Xi; \Gamma \vdash su_\varepsilon^\sigma : ?$ which implies that $\sigma := ? \in \Xi$ and $\Xi \vdash \Gamma$. Therefore, by the Rule (Rs) the result follows immediately.

Case $(t_1 t_2)$. Then, we know that

$$[t_1 t_2]_\varepsilon = \text{let } x = [t_1]_\varepsilon \text{ in let } y = [t_2]_\varepsilon \text{ in } (\varepsilon_{\rightarrow?}x :: ? \rightarrow ?) y$$

We have $t = t_1 t_2$. Then, we have to prove that

$$\mu; \Xi; \Gamma \vdash t_1 t_2 \approx \text{let } x = [t_1]_\varepsilon \text{ in let } y = [t_2]_\varepsilon \text{ in } (\varepsilon_{\rightarrow?}x :: ? \rightarrow ?) y : ?$$

Since $\Xi; \Gamma \vdash [t]_\varepsilon : ?$ and by Lemma A.137, we know that $\Xi; \Gamma \vdash [t_1]_\varepsilon : ?$ and $\Xi; \Gamma \vdash [t_2]_\varepsilon : ?$. By the induction hypothesis, we know that $\mu; \Xi; \Gamma \vdash t_1 \approx [t_1]_\varepsilon : ?$ and $\mu; \Xi; \Gamma \vdash t_2 \approx [t_2]_\varepsilon : ?$. Therefore, by the Rule (RappL) the result follows immediately.

Case $(\pi_i(t'))$. Then, we know that

$$[\pi_i(t')]_\varepsilon = \pi_i(\varepsilon_{\times?}[t']_\varepsilon :: ? \times ?)$$

We have $t = \pi_i(t')$. Then, we have to prove that $\mu; \Xi; \Gamma \vdash \pi_i(t') \approx \pi_i(\varepsilon_{\times?}[t']_\varepsilon :: ? \times ?) : ?$. Since $\Xi; \Gamma \vdash [t]_\varepsilon : ?$ and by Lemma A.137, we know that $\Xi; \Gamma \vdash [t']_\varepsilon : ?$. By the induction hypothesis, we know that $\mu; \Xi; \Gamma \vdash t' \approx [t']_\varepsilon : ?$. Therefore, by the Rule (R π_i) the result follows immediately.

Case $(\{t_1\}_{t_2})$. Then, we know that

$$[\{t_1\}_{t_2}]_\varepsilon = \text{let } x = [t_1]_\varepsilon \text{ in let } y = [t_2]_\varepsilon \text{ in } (\varepsilon_{\rightarrow?}\pi_1(\varepsilon_{\times?}y :: ? \times ?) :: ? \rightarrow ?) x$$

We have $t = \{t_1\}_{t_2}$. Then, we have to prove that

$$\mu; \Xi; \Gamma \vdash \{t_1\}_{t_2} \approx \text{let } x = [t_1]_\varepsilon \text{ in let } y = [t_2]_\varepsilon \text{ in } (\varepsilon_{\rightarrow?}\pi_1(\varepsilon_{\times?}y :: ? \times ?) :: ? \rightarrow ?) x : ?$$

Since $\Xi; \Gamma \vdash [t]_\varepsilon : ?$ and by Lemma A.137, we know that $\Xi; \Gamma \vdash [t_1]_\varepsilon : ?$ and $\Xi; \Gamma \vdash [t_2]_\varepsilon : ?$. By the induction hypothesis, we know that $\mu; \Xi; \Gamma \vdash t_1 \approx [t_1]_\varepsilon : ?$ and $\mu; \Xi; \Gamma \vdash t_2 \approx [t_2]_\varepsilon : ?$. Therefore, by the Rule (Rsd1L) the result follows immediately.

Case $(\text{let } \{x\}_{t_1} = t_2 \text{ in } t_3)$. Then, we know that

$$[\text{let } \{x\}_{t_1} = t_2 \text{ in } t_3]_\varepsilon = \text{let } x = [t_1]_\varepsilon \text{ in let } y = [t_2]_\varepsilon \text{ in let } z = \varepsilon_{\rightarrow?}\pi_2(\varepsilon_{\times?}x :: ? \times ?) :: ? \rightarrow ? y \text{ in } [t_3]_\varepsilon$$

We have $t = \text{let } \{x\}_{t_1} = t_2 \text{ in } t_3$. Then, we have to prove that

$$\mu; \Xi; \Gamma \vdash \text{let } \{x\}_{t_1} = t_2 \text{ in } t_3 \approx \text{let } x = [t_1]_\varepsilon \text{ in let } y = [t_2]_\varepsilon \text{ in let } z = \varepsilon_{\rightarrow?}\pi_2(\varepsilon_{\times?}x :: ? \times ?) :: ? \rightarrow ? y \text{ in } [t_3]_\varepsilon : ?$$

Since $\Xi; \Gamma \vdash [t]_\varepsilon : ?$ and by Lemma A.137, we know that $\Xi; \Gamma \vdash [t_1]_\varepsilon : ?$, $\Xi; \Gamma \vdash [t_2]_\varepsilon : ?$ and $\Xi; \Gamma, x : ? \vdash [t_3]_\varepsilon : ?$. By the induction hypothesis, we know that $\mu; \Xi; \Gamma \vdash t_1 \approx [t_1]_\varepsilon : ?$, $\mu; \Xi; \Gamma \vdash t_2 \approx [t_2]_\varepsilon : ?$ and $\mu; \Xi; \Gamma, x : ? \vdash t_3 \approx [t_3]_\varepsilon : ?$. Therefore, by the Rule (RunsL) the result follows immediately.

Case $(\langle t_1, t_2 \rangle)$. Then, we know that

$$[\langle t_1, t_2 \rangle]_\varepsilon = \varepsilon_{? \times ?} \langle [t_1]_\varepsilon, [t_2]_\varepsilon \rangle :: ?$$

We have $t = \langle t_1, t_2 \rangle$. Then, we have to prove that $\mu; \Xi; \Gamma \vdash \langle t_1, t_2 \rangle \approx \varepsilon_{? \times ?} \langle [t_1]_\varepsilon, [t_2]_\varepsilon \rangle :: ? : ?$. Since $\Xi; \Gamma \vdash [t]_\varepsilon : ?$ and by Lemma A.137, we know that $\Xi; \Gamma \vdash [t_1]_\varepsilon : ?$ and $\Xi; \Gamma \vdash [t_2]_\varepsilon : ?$. By the induction hypothesis, we know that $\mu; \Xi; \Gamma \vdash t_1 \approx [t_1]_\varepsilon : ?$ and $\mu; \Xi; \Gamma \vdash t_2 \approx [t_2]_\varepsilon : ?$. Therefore, by the Rule (Rpt) the result follows immediately.

Case $(op(\bar{t}'))$. Then, we know that

$$[op(\bar{t}')]_\varepsilon = \text{let } \bar{x} : ? = [\bar{t}'] \text{ in } \varepsilon_B op(\varepsilon_{\bar{B}} \bar{x} :: \bar{B}) :: ?$$

We have $t = op(\bar{t}')$. Then, we have to prove that

$$\mu; \Xi; \Gamma \vdash op(\bar{t}') \approx \text{let } \bar{x} : ? = [\bar{t}'] \text{ in } \varepsilon_B op(\varepsilon_{\bar{B}} \bar{x} :: \bar{B}) :: ? : ?$$

Since $\Xi; \Gamma \vdash [t]_\varepsilon : ?$ and by Lemma A.137, we know that $\Xi; \Gamma \vdash [\bar{t}']_\varepsilon : ?$. By the induction hypothesis, we know that $\mu; \Xi; \Gamma \vdash \bar{t}' \approx [\bar{t}']_\varepsilon : ?$. Therefore, by the Rule (Rop) the result follows immediately.

Case $(\nu x.t')$. Then, we know that

$$[\nu x.t']_\varepsilon = \text{let } x = su_\varepsilon \text{ in } [t']_\varepsilon$$

We have $t = \nu x.t'$. Then, we have to prove that $\mu; \Xi; \Gamma \vdash \nu x.t' \approx \text{let } x = su_\varepsilon \text{ in } [t']_\varepsilon : ?$. Since $\Xi; \Gamma \vdash \text{let } x = su_\varepsilon \text{ in } [t']_\varepsilon : ?$, we know that $\Xi; \Gamma, x : ? \vdash [t']_\varepsilon : ?$. By the induction hypothesis, we know that $\mu; \Xi; \Gamma, x : ? \vdash t' \approx [t']_\varepsilon : ?$. Therefore, by the Rule (RsG) the result follows immediately. □

Lemma A.127 *If $\vdash [t] \rightsquigarrow t_\varepsilon : ?$, then $\vdash t \approx t_\varepsilon : ?$.*

PROOF. Direct by A.126. □

Lemma A.128 *If $\mu; \Xi \vdash v \approx t : ?$, then $\Xi \triangleright t \mapsto^* \Xi \triangleright v'$, and $\mu; \Xi \vdash v \approx v' : ?$, for some v' .*

PROOF. The proof is a straightforward induction on the derivation of the rule $\mu; \Xi \vdash v \approx t : ?$. We only take into account rule cases where the term on the left can be a value.

Case (Rb). Trivial case because both terms in the relation are values.

$$(Rb) \frac{ty(b) = B}{\mu; \Xi \vdash b \approx \varepsilon_B b :: ? : ?}$$

Case (Rs). Trivial case because both terms in the relation are values.

$$(Rs) \frac{\sigma := ? \in \Xi}{\mu; \Xi \vdash \sigma \approx su^\sigma : ?}$$

Case (Ru).

$$(Ru) \frac{\mu; \Xi \vdash v \approx \varepsilon_D u :: ? : ?}{\mu; \Xi \vdash v \approx \varepsilon_D(\varepsilon_D u :: D) :: ? : ?}$$

If $t = \varepsilon_D(\varepsilon_D u :: D) :: ?$, then we know by the reduction rules of GSF_ε that:

$$\Xi \triangleright t \mapsto \Xi \triangleright \varepsilon_D u :: ?$$

Note that $\varepsilon_D ; \varepsilon_D = \varepsilon_D$ by Lemma A.136. Then, we have to prove that $\mu; \Xi \vdash v \approx \varepsilon_D u :: ? : ?$, which is a premise. Therefore, the result follows immediately.

Case (Rp). Trivial case because both terms in the relation are values.

$$(Rp) \frac{\mu; \Xi \vdash v_1 \approx \varepsilon_{D_1} u_1 :: ? : ? \quad \mu; \Xi \vdash v_2 \approx \varepsilon_{D_2} u_2 :: ? : ?}{\mu; \Xi \vdash \langle v_1, v_2 \rangle \approx \varepsilon_{D_1 \times D_2} \langle u_1, u_2 \rangle :: ? : ?}$$

Case (Rλ). Trivial case because both terms in the relation are values.

$$(R\lambda) \frac{\mu; \Xi; x : ? \vdash t_1 \approx t_2 : ?}{\mu; \Xi \vdash (\lambda x. t_1) \approx \varepsilon_{\gamma \rightarrow \gamma} (\lambda x. t_2) :: ? : ?}$$

Case (Rpt).

$$(Rpt) \frac{\mu; \Xi \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi \vdash t_2 \approx t'_2 : ?}{\mu; \Xi \vdash \langle t_1, t_2 \rangle \approx \varepsilon_{\gamma \times \gamma} \langle t'_1, t'_2 \rangle :: ? : ?}$$

We have $t = \varepsilon_{\gamma \times \gamma} \langle t'_1, t'_2 \rangle :: ?$. We know that $\langle t_1, t_2 \rangle = \langle v_1, v_2 \rangle$ for some v_1 and v_2 . Also, we know by premise that $\mu; \Xi \vdash v_1 \approx t'_1 : ?$ and $\mu; \Xi \vdash v_2 \approx t'_2 : ?$. Then, by the induction hypothesis, we know that exists v'_1 and v'_2 such that $\Xi \triangleright t'_1 \mapsto^* \Xi \triangleright v'_1$, $\Xi \triangleright t'_2 \mapsto^* \Xi \triangleright v'_2$, $\mu; \Xi \vdash v_1 \approx v'_1 : ?$ and $\mu; \Xi \vdash v_2 \approx v'_2 : ?$. Now, we have to prove that $\mu; \Xi \vdash \langle v_1, v_2 \rangle \approx \varepsilon_{\gamma \times \gamma} \langle v'_1, v'_2 \rangle :: ? : ?$. But the result follows immediately by the rule (Rpt).

Case (Rsed1).

$$(Rsed1) \frac{\mu; \Xi \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi \vdash t_2 \approx t'_2 : ?}{\mu; \Xi \vdash \{t_1\}_{t_2} \approx \varepsilon_{\gamma \rightarrow \gamma} \pi_1(\varepsilon_{\gamma \times \gamma} t'_1 :: ? \times ?) :: ? \rightarrow ? t'_2 : ?}$$

We have $t = \varepsilon_{\gamma \rightarrow \gamma} \pi_1(\varepsilon_{\gamma \times \gamma} t'_1 :: ? \times ?) :: ? \rightarrow ? t'_2$. Also, we know that $\{t_1\}_{t_2} = \{v\}_\sigma$, for some v and σ . Then, we know that $\mu; \Xi \vdash v \approx t'_2 : ?$ and $\mu; \Xi \vdash \sigma \approx t'_1 : ?$. Then, by the induction hypothesis, we know that exists v'_1 and v'_2 and such that $\Xi \triangleright t'_1 \mapsto^* \Xi \triangleright v'_1$, $\Xi \triangleright t'_2 \mapsto^* \Xi \triangleright v'_2$, $\mu; \Xi \vdash \sigma \approx v'_1 : ?$ and $\mu; \Xi \vdash v \approx v'_2 : ?$. By the rule (Rs), we know that $v'_1 = su_\varepsilon^\sigma$. By the dynamic semantics of GSF ε , we know that

$$\Xi \triangleright t \mapsto^* \Xi \triangleright \varepsilon_{\gamma \rightarrow \gamma} \pi_1(\varepsilon_{\gamma \times \gamma} su_\varepsilon^\sigma :: ? \times ?) :: ? \rightarrow ? v'_2 \mapsto^*$$

$$\Xi \triangleright \langle \sigma^? \rightarrow ?, ? \rightarrow ? \rangle (\lambda x : \sigma. \varepsilon_{\sigma^?} x :: ?) :: ? v'_2 \mapsto \Xi \triangleright \varepsilon_{\sigma^?} (\langle E_1, \sigma^{E_2} \rangle u :: ?) :: ? \mapsto \Xi \triangleright \langle E_1, \sigma^{E_2} \rangle u :: ?$$

where $v'_2 = \langle E_1, E_2 \rangle u :: ?$. Therefore, we have to prove that $\mu; \Xi \vdash \{v\}_\sigma \approx \langle E_1, \sigma^{E_2} \rangle u :: ? : ?$. As we know that $\mu; \Xi \vdash v \approx v'_2 : ?$ or what is the same $\mu; \Xi \vdash v \approx \langle E_1, E_2 \rangle u :: ? : ?$, by the Rule (Rsed2), the result follows immediately.

Case (Rsed1).

$$(Rsed1) \frac{\mu; \Xi; \Gamma \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma \vdash v_2 \approx v'_2 : ?}{\mu; \Xi; \Gamma \vdash \{v_1\}_{v_2} \approx \varepsilon_{\gamma \rightarrow \gamma} \pi_1(\varepsilon_{\gamma \times \gamma} v'_2 :: ? \times ?) :: ? \rightarrow ? v'_1 : ?}$$

We have $t = \varepsilon_{\gamma \rightarrow \gamma} \pi_1(\varepsilon_{\gamma \times \gamma} v'_2 :: ? \times ?) :: ? \rightarrow ? v'_1$. Also, we know that $\{v_1\}_{v_2} = \{v\}_\sigma$, for some v and σ . Then, we know that $\mu; \Xi \vdash v \approx v'_1 : ?$ and $\mu; \Xi \vdash \sigma \approx v'_2 : ?$. By the rule (Rs), we know that $v'_2 = su_\varepsilon^\sigma$. By the dynamic semantics of GSF ε , we know that

$$\Xi \triangleright t \mapsto^* \Xi \triangleright \varepsilon_{\gamma \rightarrow \gamma} \pi_1(\varepsilon_{\gamma \times \gamma} su_\varepsilon^\sigma :: ? \times ?) :: ? \rightarrow ? v'_1 \mapsto^*$$

$$\Xi \triangleright \langle \sigma^? \rightarrow ?, ? \rightarrow ? \rangle (\lambda x : \sigma. \varepsilon_{\sigma^?} x :: ?) :: ? v'_1 \mapsto \Xi \triangleright \varepsilon_{\sigma^?} (\langle E_1, \sigma^{E_2} \rangle u :: ?) :: ? \mapsto \Xi \triangleright \langle E_1, \sigma^{E_2} \rangle u :: ?$$

where $v'_1 = \langle E_1, E_2 \rangle u :: ?$. Therefore, we have to prove that $\mu; \Xi \vdash \{v\}_\sigma \approx \langle E_1, \sigma^{E_2} \rangle u :: ? : ?$. As we know that $\mu; \Xi \vdash v \approx v'_1 : ?$ or what is the same $\mu; \Xi \vdash v \approx \langle E_1, E_2 \rangle u :: ? : ?$, by the Rule (Rsed2), the result follows immediately.

Case (Rsed1L).

$$(Rsed1L) \frac{\mu; \Xi; \Gamma \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi; \Gamma \vdash t_2 \approx t'_2 : ?}{\mu; \Xi; \Gamma \vdash \{t_1\}_{t_2} \approx \text{let } x = t'_1 \text{ in let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow} \pi_1(\varepsilon_{\times} y :: ? \times ?) :: ? \rightarrow ?) x : ?}$$

We have

$$t = \text{let } x = t'_1 \text{ in let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow} \pi_1(\varepsilon_{\times} y :: ? \times ?) :: ? \rightarrow ?) x$$

Also, we know that $\{t_1\}_{t_2} = \{v\}_\sigma$, for some v and σ . Then, we know that $\mu; \Xi \vdash v \approx t'_1 : ?$ and $\mu; \Xi \vdash \sigma \approx t'_2 : ?$. Then, by the induction hypothesis, we know that exists v'_1 and v'_2 such that $\Xi \triangleright t'_1 \mapsto^* \Xi \triangleright v'_1$, $\Xi \triangleright t'_2 \mapsto^* \Xi \triangleright v'_2$, $\mu; \Xi \vdash \sigma \approx v'_2 : ?$ and $\mu; \Xi \vdash v \approx v'_1 : ?$. By the rule (Rs), we know that $v'_2 = su_\varepsilon^\sigma$. By the dynamic semantics of GSF ε , we know that

$$\begin{aligned} \Xi \triangleright t \mapsto^* \Xi \triangleright \varepsilon_{\rightarrow} \varepsilon_{\rightarrow} (\varepsilon_{\rightarrow} \pi_1(\varepsilon_{\times} su_\varepsilon^\sigma :: ? \times ?) :: ? \rightarrow ? v'_2) :: ? :: ? \mapsto^* \\ \Xi \triangleright \varepsilon_{\rightarrow} (\varepsilon_{\rightarrow} (\sigma^? \rightarrow ?, ? \rightarrow ?) (\lambda x : \sigma. \varepsilon_{\rightarrow} x :: ?) :: ? v'_2) :: ? :: ? \mapsto \\ \Xi \triangleright \varepsilon_{\rightarrow} (\varepsilon_{\rightarrow} (\varepsilon_{\rightarrow} (\langle E_1, \sigma^{E_2} \rangle u :: ?) :: ?) :: ? \mapsto \Xi \triangleright \langle E_1, \sigma^{E_2} \rangle u :: ? \end{aligned}$$

where $v'_1 = \langle E_1, E_2 \rangle u :: ?$. Therefore, we have to prove that $\mu; \Xi \vdash \{v\}_\sigma \approx \langle E_1, \sigma^{E_2} \rangle u :: ? : ?$. As we know that $\mu; \Xi \vdash v \approx v'_1 : ?$ or what is the same $\mu; \Xi \vdash v \approx \langle E_1, E_2 \rangle u :: ? : ?$, by the Rule (Rsed2), the result follows immediately.

Case (Rsed1R).

$$(Rsed1R) \frac{\mu; \Xi; \Gamma \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma \vdash t_2 \approx t'_2 : ?}{\mu; \Xi; \Gamma \vdash \{v_1\}_{t_2} \approx \text{let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow} \pi_1(\varepsilon_{\times} y :: ? \times ?) :: ? \rightarrow ?) v'_1 : ?}$$

We have

$$t = \text{let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow} \pi_1(\varepsilon_{\times} y :: ? \times ?) :: ? \rightarrow ?) v'_1$$

Also, we know that $\{v_1\}_{t_2} = \{v\}_\sigma$, for some v and σ . Then, we know that $\mu; \Xi \vdash v \approx v'_1 : ?$ and $\mu; \Xi \vdash \sigma \approx t'_2 : ?$. Then, by the induction hypothesis, we know that exists v'_2 such that $\Xi \triangleright t'_2 \mapsto^* \Xi \triangleright v'_2$ and $\mu; \Xi \vdash \sigma \approx v'_2 : ?$. By the rule (Rs), we know that $v'_2 = su_\varepsilon^\sigma$. By the dynamic semantics of GSF ε , we know that

$$\begin{aligned} \Xi \triangleright t \mapsto^* \Xi \triangleright \varepsilon_{\rightarrow} (\varepsilon_{\rightarrow} \pi_1(\varepsilon_{\times} su_\varepsilon^\sigma :: ? \times ?) :: ? \rightarrow ? v'_2) :: ? \mapsto^* \\ \Xi \triangleright \varepsilon_{\rightarrow} (\langle \sigma^? \rightarrow ?, ? \rightarrow ? \rangle (\lambda x : \sigma. \varepsilon_{\rightarrow} x :: ?) :: ? v'_2) :: ? \mapsto \\ \Xi \triangleright \varepsilon_{\rightarrow} (\varepsilon_{\rightarrow} (\varepsilon_{\rightarrow} (\langle E_1, \sigma^{E_2} \rangle u :: ?) :: ?) :: ? \mapsto \Xi \triangleright \langle E_1, \sigma^{E_2} \rangle u :: ? \end{aligned}$$

where $v'_1 = \langle E_1, E_2 \rangle u :: ?$. Therefore, we have to prove that $\mu; \Xi \vdash \{v\}_\sigma \approx \langle E_1, \sigma^{E_2} \rangle u :: ? : ?$. As we know that $\mu; \Xi \vdash v \approx v'_1 : ?$ or what is the same $\mu; \Xi \vdash v \approx \langle E_1, E_2 \rangle u :: ? : ?$, by the Rule (Rsed2), the result follows immediately.

Case (Rsed2). Trivial case because both terms in the relation are values.

$$(Rsed2) \frac{\mu; \Xi \vdash v \approx \langle E_1, E_2 \rangle u :: ? : ? \quad \sigma := ? \in \Xi}{\mu; \Xi \vdash \{v\}_\sigma \approx \langle E_1, \sigma^{E_2} \rangle u :: ? : ?}$$

Case (R?).

$$(R?) \frac{\mu; \Xi; \Gamma \vdash v \approx t' : ?}{\mu; \Xi; \Gamma \vdash v \approx \varepsilon_{\rightarrow} t' :: ? : ?}$$

We have $t = \varepsilon_{\rightarrow} t' :: ?$, where $\mu; \Xi; \Gamma \vdash v \approx t' : ?$. Then, by the induction hypothesis, we have that $\Xi \triangleright t' \mapsto^* v''$ and $\mu; \Xi; \Gamma \vdash v \approx v'' : ?$. By the dynamic semantics of GSF ε , we know that

$$\Xi \triangleright \varepsilon_{\rightarrow} t' :: ? \mapsto^* \Xi \triangleright \varepsilon_{\rightarrow} v'' :: ? \mapsto \Xi \triangleright v''$$

Therefore, the result follows immediately.

□

Lemma A.129 *If $\mu; \Xi \vdash v \approx t_\varepsilon : ?$, then there exists v_ε s.t. $\Xi \triangleright t_\varepsilon \mapsto^* \Xi \triangleright v_\varepsilon$, and $\mu; \Xi \vdash v \approx v_\varepsilon : ?$.*

PROOF. Direct by Lemma A.128. □

Lemma A.130 *If $\mu; \Xi \vdash t \approx t_* : ?$ and $t \parallel \mu \longrightarrow t' \parallel \mu'$, then $\Xi \triangleright t_* \mapsto^* \Xi' \triangleright t'_*$ and $\mu'; \Xi' \vdash t' \approx t'_* : ?$, for some t'_* .*

PROOF. The proof is a straightforward induction on $\mu; \Xi \vdash t \approx t_* : ?$ and case analysis on $t \parallel \mu \longrightarrow t' \parallel \mu'$. The following rules are the only ones that can be applied in this case.

Case (RsG).

$$\text{(RsG)} \frac{\mu; \Xi; x : ? \vdash t_1 \approx t'_1 : ?}{\mu; \Xi \vdash \nu x.t_1 \approx \text{let } x = su_\varepsilon \text{ in } t'_1 : ?}$$

Since $t \parallel \mu \longrightarrow t' \parallel \mu'$, we know that $t = \nu x.t_1$. By the reduction rules of λ_{seal} , we know that $t \parallel \mu \longrightarrow t_1[\sigma/x] \parallel \mu, \sigma$. By Lemma A.140, we know that $\Xi \triangleright su_\varepsilon \mapsto^* \Xi, \sigma := ? \triangleright su_\varepsilon^\sigma$. By Rule (Rs), we know that $\mu, \sigma; \Xi, \sigma := ? \vdash \sigma \approx su_\varepsilon^\sigma : ?$. By the reduction rules of $\text{GSF}\varepsilon$, we know that

$$\Xi \triangleright \text{let } x = su_\varepsilon \text{ in } t'_1 \mapsto^* \Xi, \sigma := ? \triangleright \text{let } x = su_\varepsilon^\sigma \text{ in } t'_1 \mapsto \Xi, \sigma := ? \triangleright \varepsilon_\gamma(t'_1[su_\varepsilon^\sigma/x]) :: ?$$

Then, we are required to show that $\mu, \sigma; \Xi, \sigma := ? \vdash t_1[\sigma/x] \approx \varepsilon_\gamma(t'_1[su_\varepsilon^\sigma/x]) :: ? : ?$. We know by the premise that $\mu; \Xi; x : ? \vdash t_1 \approx t'_1 : ?$, or what is the same $\mu, \sigma; \Xi, \sigma := ?; x : ? \vdash t_1 \approx t'_1 : ?$. Since $\mu, \sigma; \Xi, \sigma := ?; x : ? \vdash t_1 \approx t'_1 : ?$ and $\mu, \sigma; \Xi, \sigma := ? \vdash \sigma \approx su_\varepsilon^\sigma : ?$, by the Lemma A.141 and Rule (R?) the result follows immediately.

Case (Runs).

$$\text{(Runs)} \frac{\mu; \Xi \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi \vdash v_2 \approx v'_2 : ? \quad \mu; \Xi; z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi \vdash \text{let } \{z\}_{v_1} = v_2 \text{ in } t_3 \approx \text{let } z = \varepsilon_{\gamma \rightarrow ?} \pi_2(\varepsilon_{\gamma \times ?} v'_1 :: ? \times ?) :: ? \rightarrow ? v'_2 \text{ in } t'_3 : ?}$$

Since $t \parallel \mu \longrightarrow t' \parallel \mu'$, we know that $t = \text{let } \{z\}_\sigma = \{v\}_\sigma \text{ in } t_3$. By the reduction rules of λ_{seal} , we know that $t \parallel \mu \longrightarrow t_3[v/z] \parallel \mu$. We know by the premises that $\mu; \Xi \vdash \sigma \approx v'_1 : ?$ and $\mu; \Xi \vdash \{v\}_\sigma \approx \{v'\}_\sigma : ?$. Therefore, by Rules (Rs) and (Rsed2), we know that $v_1 = su_\varepsilon^\sigma$ and $v_2 = \langle E_1, \sigma^{E_2} \rangle u :: ?$, for some u, E_1 and E_2 . By the reduction rules of $\text{GSF}\varepsilon$, we know that

$$\begin{aligned} & \Xi \triangleright \text{let } z = \varepsilon_{\gamma \rightarrow ?} \pi_2(\varepsilon_{\gamma \times ?} su_\varepsilon^\sigma :: ? \times ?) :: ? \rightarrow ? (\langle E_1, \sigma^{E_2} \rangle u :: ?) \text{ in } t'_3 \mapsto^* \\ & \Xi \triangleright \text{let } z = (\langle ? \rightarrow \sigma^?, ? \rightarrow ? \rangle (\lambda x : ?. \varepsilon_{\sigma^?} x :: \sigma) :: ?) (\langle E_1, \sigma^{E_2} \rangle u :: ?) \text{ in } t'_3 \mapsto^* \\ & \Xi \triangleright \text{let } z = (\langle E_1, E_2 \rangle u :: ?) \text{ in } t'_3 \mapsto^* \Xi \triangleright t'_3[\langle E_1, E_2 \rangle u :: ?/z] \end{aligned}$$

We are required to show that $\mu; \Xi \vdash t_3[v/z] \approx t'_3[\langle E_1, E_2 \rangle u :: ?/z] : ?$, but we know that $\mu; \Xi \vdash \{v\}_\sigma \approx \langle E_1, \sigma^{E_2} \rangle u :: ? : ?$, therefore we know by the rule (Rsed2) that $\mu; \Xi \vdash v \approx \langle E_1, E_2 \rangle u :: ? : ?$. Finally, by the Lemma A.141, the result follows immediately.

Case (Rop).

$$\text{(Rop)} \frac{\mu; \Xi; \Gamma \vdash \bar{t}_1 \approx \bar{t}_2 : \bar{B} \quad ty(op) = \bar{B} \rightarrow B'}{\mu; \Xi; \Gamma \vdash op(\bar{t}_1) \approx op(\varepsilon_{\bar{B}} \bar{t}_2 :: \bar{B}) :: ? : B'}$$

Applying the induction hypothesis, reduction rules of λ_{seal} and $\text{GSF}\varepsilon$, and Rule (R δ).

Case (RunsL).

$$\text{(RunsL)} \frac{\mu; \Xi \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi \vdash t_2 \approx t'_2 : ? \quad \mu; \Xi; z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi \vdash \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 \approx \text{let } x = t'_1 \text{ in let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3 : ?}$$

Since $t \parallel \mu \longrightarrow t' \parallel \mu'$, we know that $t = \text{let } \{z\}_\sigma = \{v\}_\sigma \text{ in } t_3$. By the reduction rules of λ_{seal} , we know that $t \parallel \mu \longrightarrow t_3[v/z] \parallel \mu$. We know by the premises that $\mu; \Xi \vdash \sigma \approx t'_1 : ?$ and $\mu; \Xi \vdash \{v\}_\sigma \approx t'_2 : ?$. Therefore, by Lemma A.128, we know that $\Xi \triangleright t'_1 \mapsto^* \Xi_1 \triangleright v_1$, $\Xi \triangleright t'_2 \mapsto^* \Xi \triangleright v_2$, $\mu; \Xi \vdash \sigma \approx v_1 : ?$ and $\mu; \Xi \vdash \{v\}_\sigma \approx v_2 : ?$, for some v_1 and v_2 . By Rules (Rs) and (Rsed2), we know that $v_1 = su_\varepsilon^\sigma$ and $v_2 = \langle E_1, \sigma^{E_2} \rangle u :: ?$, for some u , E_1 and E_2 . By the reduction rules of GSF_ε , we know that

$$\begin{aligned} & \Xi \triangleright \text{let } x = t'_1 \text{ in let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3 \mapsto^* \\ & \Xi \triangleright \varepsilon_{? \varepsilon ?} (\text{let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} su_\varepsilon^\sigma :: ? \times ?) :: ? \rightarrow ? (\langle E_1, \sigma^{E_2} \rangle u :: ?) \text{ in } t'_3) :: ? :: ? \mapsto^* \\ & \Xi \triangleright \varepsilon_{? \varepsilon ?} (\text{let } z = (\langle ? \rightarrow \sigma^?, ? \rightarrow ? \rangle (\lambda x : ?. \varepsilon_{\sigma^?} x :: \sigma) :: ?) (\langle E_1, \sigma^{E_2} \rangle u :: ?) \text{ in } t'_3) :: ? :: ? \mapsto^* \\ & \Xi \triangleright \varepsilon_{? \varepsilon ?} (\text{let } z = (\langle E_1, E_2 \rangle u :: ?) \text{ in } t'_3) :: ? :: ? \mapsto^* \Xi \triangleright \varepsilon_{? \varepsilon ?} (t'_3[\langle E_1, E_2 \rangle u :: ? / x]) :: ? :: ? \mapsto^* \end{aligned}$$

We are required to show that $\mu; \Xi \vdash t_3[v/z] \approx \varepsilon_{? \varepsilon ?} (t'_3[\langle E_1, E_2 \rangle u :: ? / z]) :: ? :: ? :: ? : ?$, but we know that $\mu; \Xi \vdash \{v\}_\sigma \approx \langle E_1, \sigma^{E_2} \rangle u :: ? : ?$, therefore we know by the rule (Rsed2) that $\mu; \Xi \vdash v \approx \langle E_1, E_2 \rangle u :: ? : ?$. Finally, by the Lemma A.141 and the Rule (R?), the result follows immediately.

Case (RunsR).

$$\text{(RunsR)} \frac{\mu; \Xi \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi \vdash t_2 \approx t'_2 : ? \quad \mu; \Xi; z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi \vdash \text{let } \{z\}_{v_1} = t_2 \text{ in } t_3 \approx \text{let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1 :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3 : ?}$$

Since $t \parallel \mu \longrightarrow t' \parallel \mu'$, we know that $t = \text{let } \{z\}_\sigma = \{v\}_\sigma \text{ in } t_3$. By the reduction rules of λ_{seal} , we know that $t \parallel \mu \longrightarrow t_3[v/z] \parallel \mu$. We know by the premises that $\mu; \Xi \vdash \sigma \approx v'_1 : ?$ and $\mu; \Xi \vdash \{v\}_\sigma \approx t'_2 : ?$. Therefore, by Lemma A.128, we know that $\Xi \triangleright t'_2 \mapsto^* \Xi \triangleright v_2$ and $\mu; \Xi \vdash \{v\}_\sigma \approx v_2 : ?$, for some v_2 . By Rules (Rs) and (Rsed2), we know that $v_1 = su_\varepsilon^\sigma$ and $v_2 = \langle E_1, \sigma^{E_2} \rangle u :: ?$, for some u , E_1 and E_2 . By the reduction rules of GSF_ε , we know that

$$\begin{aligned} & \Xi \triangleright \text{let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1 :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3 \mapsto^* \\ & \Xi \triangleright \varepsilon_{?} (\text{let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} su_\varepsilon^\sigma :: ? \times ?) :: ? \rightarrow ? (\langle E_1, \sigma^{E_2} \rangle u :: ?) \text{ in } t'_3) :: ? \mapsto^* \\ & \Xi \triangleright \varepsilon_{?} (\text{let } z = (\langle ? \rightarrow \sigma^?, ? \rightarrow ? \rangle (\lambda x : ?. \varepsilon_{\sigma^?} x :: \sigma) :: ?) (\langle E_1, \sigma^{E_2} \rangle u :: ?) \text{ in } t'_3) :: ? \mapsto^* \\ & \Xi \triangleright \varepsilon_{?} (\text{let } z = (\langle E_1, E_2 \rangle u :: ?) \text{ in } t'_3) :: ? \mapsto^* \Xi \triangleright \varepsilon_{? \varepsilon ?} (t'_3[\langle E_1, E_2 \rangle u :: ? / x]) :: ? :: ? \end{aligned}$$

We are required to show that $\mu; \Xi \vdash t_3[v/z] \approx \varepsilon_{? \varepsilon ?} (t'_3[\langle E_1, E_2 \rangle u :: ? / z]) :: ? :: ? :: ? : ?$, but we know that $\mu; \Xi \vdash \{v\}_\sigma \approx \langle E_1, \sigma^{E_2} \rangle u :: ? : ?$, therefore we know by the rule (Rsed2) that $\mu; \Xi \vdash v \approx \langle E_1, E_2 \rangle u :: ? : ?$. Finally, by the Lemma A.141 and the Rule (R?), the result follows immediately.

Case (Rapp).

$$\text{(Rapp)} \frac{\mu; \Xi \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi \vdash v_2 \approx v'_2 : ?}{\mu; \Xi \vdash v_1 v_2 \approx (\varepsilon_{\rightarrow?} v'_1 :: ? \rightarrow ?) v'_2 : ?}$$

Since $t \parallel \mu \longrightarrow t' \parallel \mu'$, we know that $t = (\lambda x.t'_1) v_2$, where $v_1 = (\lambda x.t'_1)$. Therefore, we know that $\mu; \Xi \vdash (\lambda x.t'_1) \approx v'_1 : ?$ and $\mu; \Xi \vdash v_2 \approx v'_2 : ?$. By the rule (R λ), we know that $v'_1 = \varepsilon_{\rightarrow?} \lambda x.t''_1 :: ?$, where $\Xi; x : ? \vdash t''_1 \approx t'''_1 : ?$.

By the dynamic semantics of λ_{seal} , we know that

$$(\lambda x.t''_1) v_2 \parallel \mu \longrightarrow t''_1[v_2/x] \parallel \mu$$

By the dynamic semantics of $\text{GSF}\varepsilon$, we know that

$$\begin{aligned} \Xi \triangleright (\varepsilon_{\rightarrow?}(\varepsilon_{\rightarrow?} \lambda x.t''_1 :: ?) :: ? \rightarrow ?) v'_2 \mapsto^* \\ \Xi \triangleright (\varepsilon_{\rightarrow?}(\lambda x.t''_1) :: ? \rightarrow ?) v'_2 \mapsto \Xi \triangleright \varepsilon_{\rightarrow?}(t''_1[v_2/x]) :: ? \end{aligned}$$

Since $\mu; \Xi; x : ? \vdash t''_1 \approx t'''_1 : ?$ and $\mu; \Xi \vdash v_2 \approx v'_2 : ?$, we know by Lemma A.141 that $\mu; \Xi \vdash t''_1[v_2/x] \approx (t'''_1[v'_2/x]) : ?$. By the Rule (R?), we know that $\mu; \Xi \vdash t''_1[v_2/x] \approx \varepsilon_{\rightarrow?}(t'''_1[v'_2/x]) :: ? : ?$, thus the result follows.

Case (RappL).

$$\text{(RappL)} \frac{\mu; \Xi \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi \vdash t_2 \approx t'_2 : ?}{\mu; \Xi \vdash t_1 t_2 \approx \text{let } x = t'_1 \text{ in let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow?} x :: ? \rightarrow ?) y : ?}$$

Since $t \parallel \mu \longrightarrow t' \parallel \mu'$, we know that $t = (\lambda x.t'_1) v_2$, where $t_1 = (\lambda x.t'_1)$ and $t_2 = v_2$. Therefore, we know that $\mu; \Xi \vdash (\lambda x.t'_1) \approx t'_1 : ?$ and $\mu; \Xi \vdash v_2 \approx t'_2 : ?$. By Lemma A.128, we know that $\Xi \triangleright t'_1 \mapsto^* \Xi \triangleright v'_1$, $\Xi \triangleright t'_2 \mapsto^* \Xi \triangleright v'_2$, $\mu; \Xi \vdash (\lambda x.t'_1) \approx v'_1 : ?$ and $\mu; \Xi \vdash v_2 \approx v'_2 : ?$, for some v'_1 and v'_2 . By the rule (R λ), we know that $v'_1 = \varepsilon_{\rightarrow?} \lambda x.t''_1 :: ?$, where $\Xi; x : ? \vdash t''_1 \approx t'''_1 : ?$.

By the dynamic semantics of λ_{seal} , we know that

$$(\lambda x.t''_1) v_2 \parallel \mu \longrightarrow t''_1[v_2/x] \parallel \mu$$

By the dynamic semantics of $\text{GSF}\varepsilon$, we know that

$$\begin{aligned} \Xi \triangleright \text{let } x = t'_1 \text{ in let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow?} x :: ? \rightarrow ?) y \mapsto^* \Xi \triangleright \varepsilon_{\rightarrow?} \varepsilon_{\rightarrow?} \varepsilon_{\rightarrow?} (\lambda x.t''_1) :: ? \rightarrow ?) v'_2 :: ? : ? \\ \Xi \triangleright \varepsilon_{\rightarrow?} \varepsilon_{\rightarrow?} (\varepsilon_{\rightarrow?}(t''_1[v_2/x]) :: ?) :: ? : ? \end{aligned}$$

Since $\mu; \Xi; x : ? \vdash t''_1 \approx t'''_1 : ?$ and $\mu; \Xi \vdash v_2 \approx v'_2 : ?$, we know by Lemma A.141 that $\mu; \Xi \vdash t''_1[v_2/x] \approx (t'''_1[v'_2/x]) : ?$. By the Rule (R?), we know that $\mu; \Xi \vdash t''_1[v_2/x] \approx \varepsilon_{\rightarrow?}(t'''_1[v'_2/x]) :: ? : ?$, therefore we have $\mu; \Xi \vdash t''_1[v_2/x] \approx \varepsilon_{\rightarrow?} \varepsilon_{\rightarrow?} (\varepsilon_{\rightarrow?}(t'''_1[v'_2/x]) :: ?) :: ? : ? : ?$, thus the result follows.

Case (RappR).

$$\text{(RappR)} \frac{\mu; \Xi \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi \vdash t_2 \approx t'_2 : ?}{\mu; \Xi \vdash v_1 t_2 \approx \text{let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow?} v'_1 :: ? \rightarrow ?) y : ?}$$

Since $t \parallel \mu \longrightarrow t' \parallel \mu'$, we know that $t = (\lambda x.t'_1) v_2$, where $v_1 = (\lambda x.t'_1)$ and $t_2 = v_2$. Therefore, we know that $\mu; \Xi \vdash (\lambda x.t'_1) \approx v'_1 : ?$ and $\mu; \Xi \vdash v_2 \approx t'_2 : ?$. By Lemma A.128, we know

that $\Xi \triangleright t'_2 \mapsto^* \Xi \triangleright v'_2$ and $\mu; \Xi \vdash v_2 \approx v'_2 : ?$, for some v'_2 . By the rule (R λ), we know that $v'_1 = \varepsilon_{? \rightarrow ?} \lambda x. t''_1 :: ?$, where $\Xi; x : ? \vdash t''_1 \approx t'''_1 : ?$.

By the dynamic semantics of λ_{seal} , we know that

$$(\lambda x. t''_1) v_2 \parallel \mu \longrightarrow t''_1[v_2/x] \parallel \mu$$

By the dynamic semantics of GSF ε , we know that

$$\begin{aligned} \Xi \triangleright \text{let } y = t'_2 \text{ in } (\varepsilon_{? \rightarrow ?} v'_1 :: ? \rightarrow ?) y &\mapsto^* \Xi \triangleright \varepsilon_{? \varepsilon_{? \rightarrow ?}}(\lambda x. t'''_1) :: ? \rightarrow ?) v'_2) :: ? \\ \Xi \triangleright \varepsilon_{?}(\varepsilon_{?}(t'''_1[v'_2/x]) :: ?) &:: ? \end{aligned}$$

Since $\mu; \Xi; x : ? \vdash t''_1 \approx t'''_1 : ?$ and $\mu; \Xi \vdash v_2 \approx v'_2 : ?$, we know by Lemma A.141 that $\mu; \Xi \vdash t''_1[v_2/x] \approx (t'''_1[v'_2/x]) : ?$. By the Rule (R?), we know that $\mu; \Xi \vdash t''_1[v_2/x] \approx \varepsilon_{?}(t'''_1[v'_2/x]) :: ? : ?$, therefore we have $\mu; \Xi \vdash t''_1[v_2/x] \approx \varepsilon_{?}(\varepsilon_{?}(t'''_1[v'_2/x]) :: ?) :: ? : ?$, thus the result follows.

Case (Rpi).

$$\text{(Rpi)} \frac{\mu; \Xi \vdash t \approx t' : ?}{\mu; \Xi \vdash \pi_i(t) \approx \pi_i(\varepsilon_{? \times ?} t' :: ? \times ?) : ?}$$

Applying the induction hypothesis, reduction rules of λ_{seal} and GSF ε , and Rules (Rp) and (Rpt).

Case (R?). We have that

$$\text{(R?)} \frac{\mu; \Xi \vdash t \approx t_*'' : ?}{\mu; \Xi \vdash t \approx \varepsilon_{?} t_*'' :: ? : ?}$$

We have $t_* = \varepsilon_{?} t_*'' :: ?$, where $\mu; \Xi \vdash t \approx t_*'' : ?$. Then, by the induction hypothesis, we have that $\Xi \triangleright t_*'' \mapsto^* \Xi' \triangleright t_*'''$ and $\mu'; \Xi' \vdash t' \approx t_*''' : ?$. We are required to show that $\mu'; \Xi' \vdash t' \approx \varepsilon_{?} t_*''' :: ? : ?$. But the result follows immediately by the Rule (R?).

□

Lemma A.131 *Let $\mu; \Xi \vdash v_1 \approx \varepsilon u :: ? : ?$. Then, $v_1 = \lambda x. t_1$ if and only if $u = \lambda x : ?. t_2$ and $\varepsilon = \varepsilon_{? \rightarrow ?}$.*

PROOF. The proof follow by the exploration of rules in $\mu; \Xi \vdash v_1 \approx \varepsilon u :: ? : ?$ and the definition of the evidence. □

Let $\mu; \Xi \vdash v_1 \approx \varepsilon u :: ? : ?$. Then, $v_1 \neq \lambda x. t_1$ then $u \neq \lambda x : ?. t_2$ and $\varepsilon \neq \varepsilon_{G_1 \rightarrow G_2}$.

PROOF. By Lemma A.131. □

Lemma A.132 *If $\mu; \Xi \vdash t \approx t_* : ?$ and $t \parallel \mu \longrightarrow \text{error}$, then $\Xi \triangleright t_* \mapsto^* \text{error}$.*

PROOF. The proof is a straightforward induction on $\mu; \Xi \vdash t \approx t_* : ?$. The following rule is the only one that can be applied in this case ($t \parallel \mu \longrightarrow \text{error}$).

Case (Rapp).

$$\text{(Rapp)} \frac{\mu; \Xi \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi \vdash v_2 \approx v'_2 : ?}{\mu; \Xi \vdash v_1 v_2 \approx (\varepsilon_{? \rightarrow ?} v'_1 :: ? \rightarrow ?) v'_2 : ?}$$

Since $t \parallel \mu \longrightarrow \mathbf{type_error}$, we know that v_1 is not a function, and by Corollary A.9 and $\mu; \Xi \vdash v_1 \approx v'_1 : ?$, we know that v'_1 also can not be a function and its evidence, syntactically, can not be a function. Let suppose that $v'_1 = \varepsilon_1 u_1 :: ?$. Then, we know that $\varepsilon_1 \mathbin{\varepsilon}_{? \rightarrow ?}$ fails, and the result holds.

$$\Xi \triangleright (\varepsilon_{? \rightarrow ?}(\varepsilon_1 u_1 :: ?) :: ? \rightarrow ?) v'_2 \mapsto^* \mathbf{error}$$

Case (RappL).

$$\text{(RappL)} \frac{\mu; \Xi \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi \vdash t_2 \approx t'_2 : ?}{\mu; \Xi \vdash t_1 t_2 \approx \text{let } x = t'_1 \text{ in let } y = t'_2 \text{ in } (\varepsilon_{? \rightarrow ?} x :: ? \rightarrow ?) y : ?}$$

By Lemma A.128, $\mu; \Xi \vdash t_1 \approx t'_1 : ?$ and $\mu; \Xi \vdash t_2 \approx t'_2 : ?$, we know that $\Xi \triangleright t'_1 \mapsto^* \Xi \triangleright v'_1$, $\Xi \triangleright t'_2 \mapsto^* \Xi \triangleright v'_2$, $\mu; \Xi \vdash v_1 \approx v'_1 : ?$ and $\mu; \Xi \vdash v_2 \approx v'_2 : ?$, for some v'_1 and v'_2 . Since $t \parallel \mu \longrightarrow \mathbf{type_error}$, we know that v_1 is not a function, and by Corollary A.9 and $\mu; \Xi \vdash v_1 \approx v'_1 : ?$, we know that v'_1 also can not be a function and its evidence, syntactically, can not be a function. Let suppose that $v'_1 = \varepsilon_1 u_1 :: ?$. Then, we know that $\varepsilon_1 \mathbin{\varepsilon}_{? \rightarrow ?}$ fails, and the result holds.

$$\Xi \triangleright \text{let } x = t'_1 \text{ in let } y = t'_2 \text{ in } (\varepsilon_{? \rightarrow ?} x :: ? \rightarrow ?) y \mapsto^*$$

$$\Xi \triangleright \varepsilon_1 \varepsilon_{? \rightarrow ?}(\varepsilon_{? \rightarrow ?}(\varepsilon_1 u_1 :: ?) :: ? \rightarrow ?) v'_2 :: ? :: ? \mapsto^* \mathbf{error}$$

Case (RappR).

$$\text{(RappR)} \frac{\mu; \Xi \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi \vdash t_2 \approx t'_2 : ?}{\mu; \Xi \vdash v_1 t_2 \approx \text{let } y = t'_2 \text{ in } (\varepsilon_{? \rightarrow ?} v'_1 :: ? \rightarrow ?) y : ?}$$

By Lemma A.128 and $\mu; \Xi \vdash t_2 \approx t'_2 : ?$, we know that $\Xi \triangleright t'_2 \mapsto^* \Xi \triangleright v'_2$ and $\mu; \Xi \vdash v_2 \approx v'_2 : ?$, for some v'_2 . Since $t \parallel \mu \longrightarrow \mathbf{type_error}$, we know that v_1 is not a function, and by Corollary A.9 and $\mu; \Xi \vdash v_1 \approx v'_1 : ?$, we know that v'_1 also can not be a function and its evidence, syntactically, can not be a function. Let suppose that $v'_1 = \varepsilon_1 u_1 :: ?$. Then, we know that $\varepsilon_1 \mathbin{\varepsilon}_{? \rightarrow ?}$ fails, and the result holds.

$$\Xi \triangleright \text{let } y = t'_2 \text{ in } (\varepsilon_{? \rightarrow ?} v'_1 :: ? \rightarrow ?) y \mapsto^*$$

$$\Xi \triangleright \varepsilon_1 \varepsilon_{? \rightarrow ?}(\varepsilon_1 u_1 :: ?) :: ? \rightarrow ?) v'_2 :: ? \mapsto^* \mathbf{error}$$

Case (TRpi). (TRpi) $\frac{\mu; \Xi; \Gamma \vdash t \approx t' : ?}{\mu; \Xi; \Gamma \vdash \pi_i(t) \approx \pi_i(\varepsilon_{? \times ?} t' :: ? \times ?) : ?}$ Similar to the function application case.

Case (Runs).

$$\text{(Runs)} \frac{\mu; \Xi \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi \vdash v_2 \approx v'_2 : ? \quad \Xi; z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi \vdash \text{let } \{z\}_{v_1} = v_2 \text{ in } t_3 \approx \text{let } z = \varepsilon_{? \times ?} \pi_2(\varepsilon_{? \times ?} v'_1 :: ? \times ?) :: ? \rightarrow ? v'_2 \text{ in } t'_3 : ?}$$

Since $t \parallel \mu \longrightarrow \mathbf{unseal_error}$, we know that $t = \text{let } \{z\}_\sigma = \{v\}_{\sigma'} \text{ in } t_3$, where $\sigma \not\rightsquigarrow \sigma'$. We know by the premises that $\mu; \Xi \vdash \sigma \approx v'_1 : ?$ and $\mu; \Xi \vdash \{v\}_{\sigma'} \approx v'_2 : ?$. Therefore, by Rules (Rs) and

(Rsed2), we know that $v_1 = su_\varepsilon^\sigma$ and $v_2 = \langle E_1, \sigma'^{E_2} \rangle u :: ?$, for some u , E_1 and E_2 . By the reduction rules of GSF_ε , we know that

$$\begin{aligned} \Xi \triangleright \text{let } x &= \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} su_\varepsilon^\sigma :: ? \times ?) :: ? \rightarrow ? (\langle E_1, \sigma'^{E_2} \rangle u :: ?) \text{ in } t'_3 \mapsto^* \\ \Xi \triangleright \text{let } x &= (\langle ? \rightarrow \sigma^?, ? \rightarrow ? \rangle (\lambda x : ?. \varepsilon_{\sigma^?} x :: \sigma) :: ?) (\langle E_1, \sigma'^{E_2} \rangle u :: ?) \text{ in } t'_3 \mapsto^* \\ \Xi \triangleright \text{let } x &= (\langle \sigma^?, ? \rangle (\varepsilon_{\sigma^?} (\langle E_1, \sigma'^{E_2} \rangle u :: ?) :: \sigma) :: ?) \text{ in } t'_3 \mapsto \mathbf{error} \end{aligned}$$

Note that the transitivity between $\langle E_1, \sigma'^{E_2} \rangle ; \varepsilon_{\sigma^?}$ fails because $\sigma' \not\prec \sigma$. Thus the results follows immediately.

Case (RunsL).

$$\text{(RunsL)} \frac{\mu; \Xi \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi \vdash t_2 \approx t'_2 : ? \quad \mu; \Xi; z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi \vdash \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 \approx \text{let } x = t'_1 \text{ in let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3 : ?}$$

Since $t \parallel \mu \rightarrow \mathbf{unseal_error}$, we know that $t = \text{let } \{z\}_\sigma = \{v\}_{\sigma'} \text{ in } t_3$, where $\sigma \not\prec \sigma'$. We know by the premises that $\mu; \Xi \vdash \sigma \approx t'_1 : ?$ and $\mu; \Xi \vdash \{v\}_{\sigma'} \approx t'_2 : ?$. Therefore, by Lemma A.128, we know that $\Xi \triangleright t'_1 \mapsto^* \Xi \triangleright v_1$, $\Xi \triangleright t'_2 \mapsto^* \Xi \triangleright v_2$, $\mu; \Xi \vdash \sigma \approx v_1 : ?$ and $\mu; \Xi \vdash \{v\}_{\sigma'} \approx v_2 : ?$, for some v_1 and v_2 . By Rules (Rs) and (Rsed2), we know that $v_1 = su_\varepsilon^\sigma$ and $v_2 = \langle E_1, \sigma'^{E_2} \rangle u :: ?$, for some u , E_1 and E_2 . By the reduction rules of GSF_ε , we know that

$$\begin{aligned} \Xi \triangleright \text{let } x &= t'_1 \text{ in let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3 \mapsto^* \\ \Xi \triangleright \varepsilon_{? \varepsilon ?} (\text{let } z &= \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} su_\varepsilon^\sigma :: ? \times ?) :: ? \rightarrow ? (\langle E_1, \sigma'^{E_2} \rangle u :: ?) \text{ in } t'_3) :: ? :: ? \mapsto^* \\ \Xi \triangleright \varepsilon_{? \varepsilon ?} (\text{let } z &= (\langle ? \rightarrow \sigma^?, ? \rightarrow ? \rangle (\lambda x : ?. \varepsilon_{\sigma^?} x :: \sigma) :: ?) (\langle E_1, \sigma'^{E_2} \rangle u :: ?) \text{ in } t'_3) :: ? :: ? \mapsto^* \\ \Xi \triangleright \varepsilon_{? \varepsilon ?} (\text{let } x &= (\langle \sigma^?, ? \rangle (\varepsilon_{\sigma^?} (\langle E_1, \sigma'^{E_2} \rangle u :: ?) :: \sigma) :: ?) \text{ in } t'_3) :: ? :: ? \mapsto \mathbf{error} \end{aligned}$$

Note that the transitivity between $\langle E_1, \sigma'^{E_2} \rangle ; \varepsilon_{\sigma^?}$ fails because $\sigma' \not\prec \sigma$. Thus the results follows immediately.

Case (RunsR).

$$\text{(RunsR)} \frac{\mu; \Xi \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi \vdash t_2 \approx t'_2 : ? \quad \mu; \Xi; z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi \vdash \text{let } \{z\}_{v_1} = t_2 \text{ in } t_3 \approx \text{let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1 :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3 : ?}$$

Since $t \parallel \mu \rightarrow \mathbf{unseal_error}$, we know that $t = \text{let } \{z\}_\sigma = \{v\}_{\sigma'} \text{ in } t_3$, where $\sigma \not\prec \sigma'$. We know by the premises that $\mu; \Xi \vdash \sigma \approx v'_1 : ?$ and $\mu; \Xi \vdash \{v\}_{\sigma'} \approx t'_2 : ?$. Therefore, by Lemma A.128, we know that $\Xi \triangleright t'_2 \mapsto^* \Xi \triangleright v_2$ and $\mu; \Xi \vdash \{v\}_{\sigma'} \approx v_2 : ?$, for some v_2 . By Rules (Rs) and (Rsed2), we know that $v_1 = su_\varepsilon^\sigma$ and $v_2 = \langle E_1, \sigma'^{E_2} \rangle u :: ?$, for some u , E_1 and E_2 . By the reduction rules of GSF_ε , we know that

$$\begin{aligned} \Xi \triangleright \text{let } y &= t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1 :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3 \mapsto^* \\ \Xi \triangleright \varepsilon_{?} (\text{let } z &= \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} su_\varepsilon^\sigma :: ? \times ?) :: ? \rightarrow ? (\langle E_1, \sigma'^{E_2} \rangle u :: ?) \text{ in } t'_3) :: ? \mapsto^* \\ \Xi \triangleright \varepsilon_{?} (\text{let } z &= (\langle ? \rightarrow \sigma^?, ? \rightarrow ? \rangle (\lambda x : ?. \varepsilon_{\sigma^?} x :: \sigma) :: ?) (\langle E_1, \sigma'^{E_2} \rangle u :: ?) \text{ in } t'_3) :: ? \mapsto^* \\ \Xi \triangleright \varepsilon_{?} (\text{let } x &= (\langle \sigma^?, ? \rangle (\varepsilon_{\sigma^?} (\langle E_1, \sigma'^{E_2} \rangle u :: ?) :: \sigma) :: ?) \text{ in } t'_3) :: ? \mapsto \mathbf{error} \end{aligned}$$

Note that the transitivity between $\langle E_1, \sigma'^{E_2} \rangle ; \varepsilon_{\sigma^?}$ fails because $\sigma' \not\prec \sigma$. Thus the results follows immediately.

Case (R?).

$$(R?) \frac{\mu; \Xi \vdash t \approx t_{1*} : ?}{\mu; \Xi \vdash t \approx \varepsilon_{?} t_{1*} :: ? : ?}$$

Since $t \parallel \mu \longrightarrow \mathbf{error}$, we know by the induction hypothesis on $\mu; \Xi \vdash t \approx t_{1*} : ?$ that $\Xi \triangleright t_{1*} \mapsto \mathbf{error}$. Thus the result follows immediately. \square

Lemma A.133 *If $\mu; \Xi \vdash t \approx t_* : ?$ and $t \parallel \mu \mapsto t' \parallel \mu'$, then $\Xi \triangleright t_* \mapsto^* \Xi' \triangleright t'_*$ and $\mu'; \Xi' \vdash t' \approx t'_* : ?$, for some t'_* .*

PROOF. The proof is a straightforward induction on $\mu; \Xi \vdash t_1 \approx t_2 : ?$. We only take into account the rules that can be applied.

Case (Rpt).

$$(Rpt) \frac{\mu; \Xi \vdash t_1 \approx t_{1*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2*} : ?}{\mu; \Xi \vdash \langle t_1, t_2 \rangle \approx \varepsilon_{?x?} \langle t_{1*}, t_{2*} \rangle :: ? : ?}$$

If $t \parallel \mu \longrightarrow t' \parallel \mu'$, then by Lemma A.130, the result follows immediately. Else, if $t \parallel \mu \mapsto t' \parallel \mu'$, we have the following two cases:

- $t = \langle t_1, t_2 \rangle = f[t_1]$, where $f = \langle [], t_2 \rangle$.

Therefore, we have that $t_1 \parallel \mu \mapsto t'_1 \parallel \mu'$.

By the induction hypothesis, we get that $\Xi \triangleright t_{1*} \mapsto^* \Xi' \triangleright t'_{1*}$ and $\mu'; \Xi' \vdash t'_1 \approx t'_{1*} : ?$. Thus, we know that

$$\Xi \triangleright \varepsilon_{?x?} \langle t_{1*}, t_{2*} \rangle :: ? \mapsto^* \Xi' \triangleright \varepsilon_{?x?} \langle t'_{1*}, t_{2*} \rangle :: ?$$

Therefore, the result follows immediately by Rule (Rpt).

- $t = \langle t_1, t_2 \rangle = \langle v_1, t_2 \rangle = f[t_2]$, where $f = \langle v_1, [] \rangle$. Therefore, we have that $t_2 \parallel \mu \mapsto t'_2 \parallel \mu'$. By the induction hypothesis, we get that $\Xi \triangleright t_{2*} \mapsto^* \Xi' \triangleright t'_{2*}$ and $\mu'; \Xi' \vdash t'_2 \approx t'_{2*} : ?$. Since $\mu; \Xi \vdash v_1 \approx t_{1*} : ?$, by Lemma A.128, we know that $\Xi \triangleright t_{1*} \mapsto^* \Xi \triangleright v_{1*}$ and $\mu; \Xi \vdash v_1 \approx v_{1*} : ?$. Thus, we know that

$$\Xi \triangleright \varepsilon_{?x?} \langle t_{1*}, t_{2*} \rangle :: ? \mapsto^* \Xi \triangleright \varepsilon_{?x?} \langle v_{1*}, t_{2*} \rangle :: ? \mapsto^* \Xi' \triangleright \varepsilon_{?x?} \langle v'_{1*}, t'_{2*} \rangle :: ?$$

Therefore, the result follows immediately by Rule (Rpt).

Case (R?).

$$(R?) \frac{\mu; \Xi \vdash t_1 \approx t_{1*} : ?}{\mu; \Xi \vdash t_1 \approx \varepsilon_{?} t_{1*} :: ? : ?}$$

If $t \parallel \mu \longrightarrow t' \parallel \mu'$, then by Lemma A.130, the result follows immediately. Else, if $t \parallel \mu \mapsto t' \parallel \mu'$, we have that $t_1 \parallel \mu \mapsto t'_1 \parallel \mu'$.

By the induction hypothesis, we get that $\Xi \triangleright t_{1*} \mapsto^* \Xi' \triangleright t'_{1*}$ and $\mu'; \Xi' \vdash t'_1 \approx t'_{1*} : ?$. Thus, we know that

$$\Xi \triangleright \varepsilon_{?} t_{1*} :: ? \mapsto^* \Xi' \triangleright \varepsilon_{?} t'_{1*} :: ?$$

Therefore, the result follows immediately by Rule (R?).

Case (RappL).

$$\text{(RappL)} \frac{\mu; \Xi \vdash t_1 \approx t_{1*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2*} : ?}{\mu; \Xi \vdash t_1 t_2 \approx \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} x :: ? \rightarrow ?) y : ?}$$

If $t \parallel \mu \longrightarrow t' \parallel \mu'$, then by Lemma A.130, the result follows immediately. Else, if $t \parallel \mu \mapsto t' \parallel \mu'$, we have the following two cases:

- $t = t_1 t_2 = f[t_1]$, where $f = [] t_2$. Therefore, we have that $t_1 \parallel \mu \mapsto t_1 \parallel \mu'$.

By the induction hypothesis, we get that $\Xi \triangleright t_{1*} \mapsto^* \Xi' \triangleright t'_{1*}$ and $\mu'; \Xi' \vdash t'_1 \approx t'_{1*} : ?$. Thus, we know that

$$\begin{aligned} \Xi \triangleright \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} x :: ? \rightarrow ?) y &\mapsto^* \\ \Xi' \triangleright \text{let } x = t'_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} x :: ? \rightarrow ?) y & \end{aligned}$$

Therefore, the result follows immediately by Rule (RappL).

- $t = t_1 t_2 = v_1 t_2 = f[t_2]$, where $f = v_1 []$. Therefore, we have that $t_2 \parallel \mu \mapsto t'_2 \parallel \mu'$. By the induction hypothesis, we get that $\Xi \triangleright t_{2*} \mapsto^* \Xi' \triangleright t'_{2*}$ and $\mu'; \Xi' \vdash t'_2 \approx t'_{2*} : ?$. Since $\mu; \Xi \vdash v_1 \approx t_{1*} : ?$, by Lemma A.128, we know that $\Xi \triangleright t_{1*} \mapsto^* \Xi \triangleright v_{1*}$ and $\mu; \Xi \vdash v_1 \approx v_{1*} : ?$. Thus, we know that

$$\begin{aligned} \Xi \triangleright \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} x :: ? \rightarrow ?) y &\mapsto^* \\ \Xi \triangleright \text{let } x = v_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} x :: ? \rightarrow ?) y &\mapsto \\ \Xi \triangleright \varepsilon_{? \rightarrow ?}(\text{let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} v_{1*} :: ? \rightarrow ?) y) :: ? &\mapsto^* \\ \Xi' \triangleright \varepsilon_{? \rightarrow ?}(\text{let } y = t'_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} v_{1*} :: ? \rightarrow ?) y) :: ? & \end{aligned}$$

Therefore, the result follows immediately by Rules (RappR) and (R?).

Case (RappR).

$$\text{(RappR)} \frac{\mu; \Xi \vdash v_1 \approx v_{1*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2*} : ?}{\mu; \Xi \vdash v_1 t_2 \approx \text{let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} v_{1*} :: ? \rightarrow ?) y : ?}$$

If $t \parallel \mu \longrightarrow t' \parallel \mu'$, then by Lemma A.130, the result follows immediately. Else, if $t \parallel \mu \mapsto t' \parallel \mu'$, we know that $t = v_1 t_2 = f[t_2]$, where $f = v_1 []$. Therefore, we have that $t_2 \parallel \mu \mapsto t'_2 \parallel \mu'$. By the induction hypothesis, we get that $\Xi \triangleright t_{2*} \mapsto^* \Xi' \triangleright t'_{2*}$ and $\mu'; \Xi' \vdash t'_2 \approx t'_{2*} : ?$. Thus, we know that

$$\begin{aligned} \Xi \triangleright \text{let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} v_{1*} :: ? \rightarrow ?) y &\mapsto^* \\ \Xi' \triangleright \text{let } y = t'_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} v_{1*} :: ? \rightarrow ?) y & \end{aligned}$$

Therefore, the result follows immediately by Rule (RappR).

Case (Rpi).

$$\text{(Rpi)} \frac{\mu; \Xi \vdash t_1 \approx t_{1*} : ?}{\mu; \Xi \vdash \pi_i(t_1) \approx \pi_i(\varepsilon_{? \times ?} t_{1*} :: ? \times ?) : ?}$$

If $t \parallel \mu \longrightarrow t' \parallel \mu'$, then by Lemma A.130, the result follows immediately. Else, if $t \parallel \mu \mapsto t' \parallel \mu'$, we know that $t = \pi_i(t_1) = f[t_1]$, where $\pi_i([])$.

Therefore, we have that $t_1 \parallel \mu \mapsto t'_1 \parallel \mu'$.

By the induction hypothesis, we get that $\Xi \triangleright t_{1*} \mapsto^* \Xi' \triangleright t'_{1*}$ and $\mu'; \Xi' \vdash t'_1 \approx t'_{1*} : ?$. Thus, we know that

$$\Xi \triangleright \pi_i(\varepsilon_{? \times ?} t_{1*} :: ? \times ?) \mapsto^* \Xi' \triangleright \pi_i(\varepsilon_{? \times ?} t'_{1*} :: ? \times ?)$$

Therefore, the result follows immediately by Rule (Rpi).

Case (Rsed1L).

$$\text{(Rsed1L)} \frac{\mu; \Xi \vdash t_1 \approx t_{1*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2*} : ?}{\mu; \Xi \vdash \{t_1\}_{t_2} \approx \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) x : ?}$$

If $t \parallel \mu \longrightarrow t' \parallel \mu'$, then by Lemma A.130, the result follows immediately. Else, if $t \parallel \mu \mapsto t' \parallel \mu'$, we have the following two cases:

- $t = \{t_1\}_{t_2} = f[t_1]$, where $f = \{\square\}_{t_2}$.

Therefore, we have that $t_1 \parallel \mu \mapsto t'_1 \parallel \mu'$.

By the induction hypothesis, we get that $\Xi \triangleright t_{1*} \mapsto^* \Xi' \triangleright t'_{1*}$ and $\mu'; \Xi' \vdash t'_1 \approx t'_{1*} : ?$. Thus, we know that

$$\begin{aligned} & \Xi \triangleright \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) x \mapsto^* \\ & \Xi' \triangleright \text{let } x = t'_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) x \end{aligned}$$

Therefore, the result follows immediately by Rule (Rsed1L).

- $t = \{t_1\}_{t_2} = \{v_1\}_{t_2} = f[t_2]$, where $f = \{v_1\}_{\square}$. Therefore, we have that $t_2 \parallel \mu \mapsto t'_2 \parallel \mu'$. By the induction hypothesis, we get that $\Xi \triangleright t_{2*} \mapsto^* \Xi' \triangleright t'_{2*}$ and $\mu'; \Xi' \vdash t'_2 \approx t'_{2*} : ?$. Since $\mu; \Xi \vdash v_1 \approx t_{1*} : ?$, by Lemma A.128, we know that $\Xi \triangleright t_{1*} \mapsto^* \Xi \triangleright v_{1*}$ and $\mu; \Xi \vdash v_1 \approx v_{1*} : ?$. Thus, we know that

$$\begin{aligned} & \Xi \triangleright \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) x \mapsto^* \\ & \Xi \triangleright \text{let } x = v'_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) x \mapsto \\ & \Xi \triangleright \varepsilon_{\rightarrow?}(\text{let } y = t_{2*} \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) v'_{1*}) :: ? \mapsto^* \\ & \Xi' \triangleright \varepsilon_{\rightarrow?}(\text{let } y = t'_{2*} \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) v'_{1*}) :: ? \end{aligned}$$

Therefore, the result follows immediately by Rules (Rsed1R) and (R?).

Case (Rsed1R).

$$\text{(Rsed1R)} \frac{\mu; \Xi \vdash v_1 \approx v_{1*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2*} : ?}{\mu; \Xi \vdash \{v_1\}_{t_2} \approx \text{let } y = t_{2*} \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) v_{1*} : ?}$$

If $t \parallel \mu \longrightarrow t' \parallel \mu'$, then by Lemma A.130, the result follows immediately. Else, if $t \parallel \mu \mapsto t' \parallel \mu'$, we know that $t = \{t_1\}_{t_2} = \{v_1\}_{t_2} = f[t_2]$, where $f = \{v_1\}_{\square}$. Therefore, we have that $t_2 \parallel \mu \mapsto t'_2 \parallel \mu'$. By the induction hypothesis, we get that $\Xi \triangleright t_{2*} \mapsto^* \Xi' \triangleright t'_{2*}$ and $\mu'; \Xi' \vdash t'_2 \approx t'_{2*} : ?$. Thus, we know that

$$\begin{aligned} & \Xi \triangleright \text{let } y = t_{2*} \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) v'_{1*} \mapsto^* \\ & \Xi' \triangleright \text{let } y = t'_{2*} \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) v'_{1*} \end{aligned}$$

Therefore, the result follows immediately by Rule (Rsed1R).

Case (RunsL).

$$\text{(RunsL)} \frac{\mu; \Xi \vdash t_1 \approx t_{1*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2*} : ? \quad \mu; \Xi; z : ? \vdash t_3 \approx t_{3*} : ?}{\mu; \Xi \vdash \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 \approx \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_{3*} : ?}$$

If $t \parallel \mu \longrightarrow t' \parallel \mu'$, then by Lemma A.130, the result follows immediately. Else, if $t \parallel \mu \mapsto t' \parallel \mu'$, we have the following two cases:

- $t = \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 = f[t_1]$, where $f = \text{let } \{z\}_{\square} = t_2 \text{ in } t_3$.

Therefore, we have that $t_1 \parallel \mu \mapsto t'_1 \parallel \mu'$.

By the induction hypothesis, we get that $\Xi \triangleright t_{1^*} \mapsto^* \Xi' \triangleright t'_{1^*}$ and $\mu'; \Xi' \vdash t'_1 \approx t'_{1^*} : ?$. Thus, we know that

$$\Xi \triangleright \text{let } x = t_{1^*} \text{ in let } y = t_{2^*} \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_{3^*} \mapsto^*$$

$$\Xi' \triangleright \text{let } x = t'_{1^*} \text{ in let } y = t_{2^*} \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_{3^*}$$

Therefore, the result follows immediately by Rule (RunsL).

- $t = \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 = \text{let } \{z\}_{v_1} = t_2 \text{ in } t_3 = f[t_2]$, where $f = \text{let } \{z\}_{v_1} = \square \text{ in } t_3$. Therefore, we have that $t_2 \parallel \mu \mapsto t'_2 \parallel \mu'$. By the induction hypothesis, we get that $\Xi \triangleright t_{2^*} \mapsto^* \Xi' \triangleright t'_{2^*}$ and $\mu'; \Xi' \vdash t'_2 \approx t'_{2^*} : ?$. Since $\mu; \Xi \vdash v_1 \approx t_{1^*} : ?$, by Lemma A.128, we know that $\Xi \triangleright t_{1^*} \mapsto^* \Xi \triangleright v_{1^*}$ and $\mu; \Xi \vdash v_1 \approx v_{1^*} : ?$. Thus, we know that

$$\Xi \triangleright \text{let } x = t_{1^*} \text{ in let } y = t_{2^*} \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_{3^*} \mapsto^*$$

$$\Xi \triangleright \text{let } x = v_{1^*} \text{ in let } y = t_{2^*} \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_{3^*} \mapsto$$

$$\Xi \triangleright \varepsilon_{\rightarrow?}(\text{let } y = t_{2^*} \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} v_{1^*} :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_{3^*}) :: ? \mapsto^*$$

$$\Xi' \triangleright \varepsilon_{\rightarrow?}(\text{let } y = t'_{2^*} \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} v_{1^*} :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_{3^*}) :: ?$$

Therefore, the result follows immediately by Rules (RunsR) and (R?).

Case (RunsR).

$$\text{(RunsR)} \frac{\mu; \Xi \vdash v_1 \approx v_{1^*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2^*} : ? \quad \mu; \Xi; z : ? \vdash t_3 \approx t_{3^*} : ?}{\mu; \Xi \vdash \text{let } \{z\}_{v_1} = t_2 \text{ in } t_3 \approx \text{let } y = t_{2^*} \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} v_{1^*} :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_{3^*} : ?}$$

If $t \parallel \mu \mapsto t' \parallel \mu'$, then by Lemma A.130, the result follows immediately. Else, if $t \parallel \mu \mapsto t' \parallel \mu'$, then we know that $t = \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 = \text{let } \{z\}_{v_1} = t_2 \text{ in } t_3 = f[t_2]$, where $f = \text{let } \{z\}_{v_1} = \square \text{ in } t_3$. Therefore, we have that $t_2 \parallel \mu \mapsto t'_2 \parallel \mu'$. By the induction hypothesis, we get that $\Xi \triangleright t_{2^*} \mapsto^* \Xi' \triangleright t'_{2^*}$ and $\mu'; \Xi' \vdash t'_2 \approx t'_{2^*} : ?$. Thus, we know that

$$\Xi \triangleright \text{let } y = t_{2^*} \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} v_{1^*} :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_{3^*} \mapsto^*$$

$$\Xi' \triangleright \text{let } y = t'_{2^*} \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} v_{1^*} :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_{3^*}$$

Therefore, the result follows immediately by Rule (RunsR).

Case (RsG).

$$\text{(RsG)} \frac{\Xi; \Gamma, x : ? \vdash t_1 \approx t'_1 : ?}{\mu; \Xi \vdash \nu x. t_1 \approx \text{let } x = su_\varepsilon \text{ in } t'_1 : ?}$$

Since $t = \nu x. t_1$, we know that $t \parallel \mu \mapsto t' \parallel \mu'$. Therefore, by Lemma A.130, the result follows immediately.

Case (Runs).

$$\text{(Runs)} \frac{\mu; \Xi \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi \vdash v_2 \approx v'_2 : ? \quad \mu; \Xi; z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi \vdash \text{let } \{z\}_{v_1} = v_2 \text{ in } t_3 \approx \text{let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} v'_1 :: ? \times ?) :: ? \rightarrow ? v'_2 \text{ in } t'_3 : ?}$$

Since $t = \text{let } \{z\}_{v_1} = v_2 \text{ in } t_3$, we know that $t \parallel \mu \mapsto t' \parallel \mu'$. Therefore, by Lemma A.130, the result follows immediately.

Case (Rapp).

$$\text{(Rapp)} \frac{\mu; \Xi \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi \vdash v_2 \approx v'_2 : ?}{\mu; \Xi \vdash v_1 v_2 \approx (\varepsilon_{\rightarrow?} v'_1 :: ? \rightarrow ?) v'_2 : ?}$$

Since $t = v_1 v_2$, we know that $t \parallel \mu \longrightarrow t' \parallel \mu'$. Therefore, by Lemma A.130, the result follows immediately. \square

Lemma A.134 *If $\mu; \Xi \vdash t_1 \approx t_2 : ?$, $\Xi \subseteq \Xi'$ and $\Gamma \subseteq \Gamma'$, then $\Xi'; \Gamma' \vdash t_1 \approx t_2 : ?$.*

PROOF. The proof is a straightforward induction on $\mu; \Xi \vdash t_1 \approx t_2 : ?$. \square

Lemma A.135 *If $\mu; \Xi \vdash t \approx t_* : ?$ and $t \parallel \mu \mapsto \mathbf{error}$, then $\Xi \triangleright t \mapsto^* \mathbf{error}$.*

PROOF. The proof is a straightforward induction on $\mu; \Xi \vdash t_1 \approx t_2 : ?$. We only take into account the rules that can be applied ($t \parallel \mu \mapsto \mathbf{error}$).

Case (Rpt).

$$\text{(Rpt)} \frac{\mu; \Xi \vdash t_1 \approx t_{1*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2*} : ?}{\mu; \Xi \vdash \langle t_1, t_2 \rangle \approx \varepsilon_{\times?} \langle t_{1*}, t_{2*} \rangle :: ? : ?}$$

If $t \parallel \mu \mapsto \mathbf{error}$, then by Lemma A.132, the result follows immediately. Else, if

$$t \parallel \mu \mapsto \mathbf{error}$$

, we have the following two cases:

- $t = \langle t_1, t_2 \rangle = f[t_1]$, where $f = \langle \square, t_2 \rangle$. Therefore, we have that $t_1 \parallel \mu \mapsto \mathbf{error}$. By the induction hypothesis, we get that $\Xi \triangleright t_{1*} \mapsto^* \mathbf{error}$. Therefore, the result follows immediately.
- $t = \langle t_1, t_2 \rangle = \langle v_1, t_2 \rangle = f[t_2]$, where $f = \langle v_1, \square \rangle$.

Therefore, we have that $t_2 \parallel \mu \mapsto \mathbf{error}$. By the induction hypothesis, we get that $\Xi \triangleright t_{2*} \mapsto^* \mathbf{error}$. Since $\mu; \Xi \vdash v_1 \approx t_{1*} : ?$, by Lemma A.128, we know that $\Xi \triangleright t_{1*} \mapsto^* \Xi \triangleright v_{1*}$ and $\mu; \Xi \vdash v_1 \approx v_{1*} : ?$. Thus, we know that

$$\Xi \triangleright \varepsilon_{\times?} \langle t_{1*}, t_{2*} \rangle :: ? \mapsto^* \Xi \triangleright \varepsilon_{\times?} \langle v'_{1*}, t_{2*} \rangle :: ? \mapsto^* \mathbf{error}$$

Therefore, the result follows immediately.

Case (R?).

$$\text{(R?)} \frac{\mu; \Xi \vdash t_1 \approx t_{1*} : ?}{\mu; \Xi \vdash t_1 \approx \varepsilon_{?} t_{1*} :: ? : ?}$$

If $t \parallel \mu \mapsto \mathbf{error}$, then by Lemma A.132, the result follows immediately. Else, if

$$t \parallel \mu \mapsto \mathbf{error}$$

, we have that $t_1 \parallel \mu \mapsto \mathbf{error}$. By the induction hypothesis, we get that $\Xi \triangleright t_{1*} \mapsto^* \mathbf{error}$. Thus, we know that

$$\Xi \triangleright \varepsilon_{?} t_{1*} :: ? \mapsto^* \mathbf{error}$$

Therefore, the result follows immediately.

Case (RappL).

$$\text{(RappL)} \frac{\mu; \Xi \vdash t_1 \approx t_{1*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2*} : ?}{\mu; \Xi \vdash t_1 t_2 \approx \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} x :: ? \rightarrow ?) y : ?}$$

If $t \parallel \mu \longrightarrow \mathbf{error}$, then by Lemma A.132, the result follows immediately. Else, if

$$t \parallel \mu \mapsto \mathbf{error}$$

, we have the following two cases:

- $t = t_1 t_2 = f[t_1]$, where $f = [] t_2$. Therefore, we have that $t_1 \parallel \mu \mapsto \mathbf{error}$. By the induction hypothesis, we get that $\Xi \triangleright t_{1*} \mapsto^* \mathbf{error}$. Thus, we know that

$$\Xi \triangleright \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} x :: ? \rightarrow ?) y \mapsto^* \mathbf{error}$$

Therefore, the result follows immediately by Rule (RappL).

- $t = t_1 t_2 = v_1 t_2 = f[t_2]$, where $f = v_1 []$.

Therefore, we have that $t_2 \parallel \mu \mapsto \mathbf{error}$. By the induction hypothesis, we get that $\Xi \triangleright t_{2*} \mapsto^* \mathbf{error}$. Since $\mu; \Xi \vdash v_1 \approx t_{1*} : ?$, by Lemma A.128, we know that $\Xi \triangleright t_{1*} \mapsto^* \Xi \triangleright v_{1*}$ and $\mu; \Xi \vdash v_1 \approx v_{1*} : ?$. Thus, we know that

$$\begin{aligned} & \Xi \triangleright \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} x :: ? \rightarrow ?) y \mapsto^* \\ & \Xi \triangleright \text{let } x = v_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} x :: ? \rightarrow ?) y \mapsto \\ & \Xi \triangleright \varepsilon_{?}(\text{let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} v_{1*} :: ? \rightarrow ?) y) :: ? \mapsto^* \mathbf{error} \end{aligned}$$

Therefore, the result follows immediately.

Case (RappR).

$$\text{(RappR)} \frac{\mu; \Xi \vdash v_1 \approx v_{1*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2*} : ?}{\mu; \Xi \vdash v_1 t_2 \approx \text{let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} v_{1*} :: ? \rightarrow ?) y : ?}$$

If $t \parallel \mu \longrightarrow \mathbf{error}$, then by Lemma A.132, the result follows immediately. Else, if

$$t \parallel \mu \mapsto \mathbf{error}$$

, we know that $t = v_1 t_2 = f[t_2]$, where $f = v_1 []$.

Therefore, we have that $t_2 \parallel \mu \mapsto \mathbf{error}$. By the induction hypothesis, we get that $\Xi \triangleright t_{2*} \mapsto^* \mathbf{error}$. Thus, we know that

$$\Xi \triangleright \text{let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} v_{1*} :: ? \rightarrow ?) y \mapsto^* \mathbf{error}$$

Therefore, the result follows immediately.

Case (Rpi).

$$\text{(Rpi)} \frac{\mu; \Xi \vdash t_1 \approx t_{1*} : ?}{\mu; \Xi \vdash \pi_i(t_1) \approx \pi_i(\varepsilon_{? \times ?} t_{1*} :: ? \times ?) : ?}$$

If $t \parallel \mu \longrightarrow t' \parallel \mu'$, then by Lemma A.130, the result follows immediately. Else, if $t \parallel \mu \mapsto t' \parallel \mu'$, we know that $t = \pi_i(t_1) = f[t_1]$, where $\pi_i([])$.

Therefore, we have that $t_1 \parallel \mu \mapsto t'_1 \parallel \mu'$.

By the induction hypothesis, we get that $\Xi \triangleright t_{1*} \mapsto^* \Xi' \triangleright t'_{1*}$ and $\mu'; \Xi' \vdash t'_1 \approx t'_{1*} : ?$. Thus, we know that

$$\Xi \triangleright \pi_i(\varepsilon_{? \times ?} t_{1*} :: ? \times ?) \mapsto^* \Xi' \triangleright \pi_i(\varepsilon_{? \times ?} t'_{1*} :: ? \times ?)$$

Therefore, the result follows immediately by Rule (Rpi).

Case (Rsed1L).

$$\text{(Rsed1L)} \frac{\mu; \Xi \vdash t_1 \approx t_{1*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2*} : ?}{\mu; \Xi \vdash \{t_1\}_{t_2} \approx \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} \pi_1(\varepsilon_{? \times ?} y :: ? \times ?) :: ? \rightarrow ?) x : ?}$$

If $t \parallel \mu \longrightarrow \mathbf{error}$, then by Lemma A.132, the result follows immediately. Else, if

$$t \parallel \mu \mapsto \mathbf{error}$$

, we have the following two cases:

- $t = \{t_1\}_{t_2} = f[t_1]$, where $f = \{\square\}_{t_2}$. Therefore, we have that $t_1 \parallel \mu \mapsto \mathbf{error}$. By the induction hypothesis, we get that $\Xi \triangleright t_{1*} \mapsto^* \mathbf{error}$. Thus, we know that

$$\Xi \triangleright \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} \pi_1(\varepsilon_{? \times ?} y :: ? \times ?) :: ? \rightarrow ?) x \mapsto^* \mathbf{error}$$

Therefore, the result follows immediately.

- $t = \{t_1\}_{t_2} = \{v_1\}_{t_2} = f[t_2]$, where $f = \{v_1\}_{\square}$.

Therefore, we have that $t_2 \parallel \mu \mapsto \mathbf{error}$. By the induction hypothesis, we get that $\Xi \triangleright t_{2*} \mapsto^* \mathbf{error}$. Since $\mu; \Xi \vdash v_1 \approx t_{1*} : ?$, by Lemma A.128, we know that $\Xi \triangleright t_{1*} \mapsto^* \Xi \triangleright v_{1*}$ and $\mu; \Xi \vdash v_1 \approx v_{1*} : ?$. Thus, we know that

$$\begin{aligned} & \Xi \triangleright \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} \pi_1(\varepsilon_{? \times ?} y :: ? \times ?) :: ? \rightarrow ?) x \mapsto^* \\ & \Xi \triangleright \text{let } x = v'_{1*} \text{ in let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} \pi_1(\varepsilon_{? \times ?} y :: ? \times ?) :: ? \rightarrow ?) x \mapsto \\ & \Xi \triangleright \varepsilon_{?}(\text{let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} \pi_1(\varepsilon_{? \times ?} y :: ? \times ?) :: ? \rightarrow ?) v'_{1*}) :: ? \mapsto^* \mathbf{error} \end{aligned}$$

Therefore, the result follows immediately.

Case (Rsed1R).

$$\text{(Rsed1R)} \frac{\mu; \Xi \vdash v_1 \approx v_{1*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2*} : ?}{\mu; \Xi \vdash \{v_1\}_{t_2} \approx \text{let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} \pi_1(\varepsilon_{? \times ?} y :: ? \times ?) :: ? \rightarrow ?) v_{1*} : ?}$$

If $t \parallel \mu \longrightarrow \mathbf{error}$, then by Lemma A.132, the result follows immediately. Else, if

$$t \parallel \mu \mapsto \mathbf{error}$$

, we know that $t = \{t_1\}_{t_2} = \{v_1\}_{t_2} = f[t_2]$, where $f = \{v_1\}_{\square}$. Therefore, we have that

$$t_2 \parallel \mu \mapsto \mathbf{error}$$

By the induction hypothesis, we get that $\Xi \triangleright t_{2*} \mapsto^* \mathbf{error}$. Thus, we know that

$$\Xi \triangleright \text{let } y = t_{2*} \text{ in } (\varepsilon_{? \rightarrow ?} \pi_1(\varepsilon_{? \times ?} y :: ? \times ?) :: ? \rightarrow ?) v'_{1*} \mapsto^* \mathbf{error}$$

Therefore, the result follows immediately.

Case (RunsL).

$$\text{(RunsL)} \frac{\mu; \Xi \vdash t_1 \approx t_{1*} : ? \quad \mu; \Xi \vdash t_2 \approx t_{2*} : ? \quad \mu; \Xi; z : ? \vdash t_3 \approx t_{3*} : ?}{\mu; \Xi \vdash \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 \approx \text{let } x = t_{1*} \text{ in let } y = t_{2*} \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_{3*} : ?}$$

If $t \parallel \mu \longrightarrow \mathbf{error}$, then by Lemma A.132, the result follows immediately. Else, if

$$t \parallel \mu \mapsto \mathbf{error}$$

, we have the following two cases:

- $t = \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 = f[t_1]$, where $f = \text{let } \{z\}_{\square} = t_2 \text{ in } t_3$. Therefore, we have that

$$t_1 \parallel \mu \mapsto \mathbf{error}$$

By the induction hypothesis, we get that $\Xi \triangleright t_1^* \mapsto^* \mathbf{error}$. Thus, we know that

$$\Xi \triangleright \text{let } x = t_1^* \text{ in let } y = t_2^* \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_3^* \mapsto^* \mathbf{error}$$

Therefore, the result follows immediately.

- $t = \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 = \text{let } \{z\}_{v_1} = t_2 \text{ in } t_3 = f[t_2]$, where $f = \text{let } \{z\}_{v_1} = \square \text{ in } t_3$. Therefore, we have that

$$t_2 \parallel \mu \mapsto \mathbf{error}$$

By the induction hypothesis, we get that $\Xi \triangleright t_2^* \mapsto^* \mathbf{error}$. Since $\mu; \Xi \vdash v_1 \approx t_1^* : ?$, by Lemma A.128, we know that $\Xi \triangleright t_1^* \mapsto^* \Xi \triangleright v_1^*$ and $\mu; \Xi \vdash v_1 \approx v_1^* : ?$. Thus, we know that

$$\Xi \triangleright \text{let } x = t_1^* \text{ in let } y = t_2^* \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_3^* \mapsto^*$$

$$\Xi \triangleright \text{let } x = v_1^* \text{ in let } y = t_2^* \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} x :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_3^* \mapsto$$

$$\Xi \triangleright \varepsilon_{\rightarrow?}(\text{let } y = t_2^* \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} v_1^* :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_3^*) :: ? \mapsto^* \mathbf{error}$$

Therefore, the result follows immediately.

Case (RunsR).

$$\text{(RunsR)} \frac{\mu; \Xi \vdash v_1 \approx v_1^* : ? \quad \mu; \Xi \vdash t_2 \approx t_2^* : ? \quad \mu; \Xi; z : ? \vdash t_3 \approx t_3^* : ?}{\mu; \Xi \vdash \text{let } \{z\}_{v_1} = t_2 \text{ in } t_3 \approx \text{let } y = t_2^* \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} v_1^* :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_3^* : ?}$$

If $t \parallel \mu \mapsto \mathbf{error}$, then by Lemma A.132, the result follows immediately. Else, if

$$t \parallel \mu \mapsto \mathbf{error}$$

, then we know that $t = \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 = \text{let } \{z\}_{v_1} = t_2 \text{ in } t_3 = f[t_2]$, where $f = \text{let } \{z\}_{v_1} = \square \text{ in } t_3$. Therefore, we have that

$$t_2 \parallel \mu \mapsto \mathbf{error}$$

By the induction hypothesis, we get that $\Xi \triangleright t_2^* \mapsto^* \mathbf{error}$. Thus, we know that

$$\Xi \triangleright \text{let } y = t_2^* \text{ in let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} v_1^* :: ? \times ?) :: ? \rightarrow ? y \text{ in } t_3^* \mapsto^* \mathbf{error}$$

Therefore, the result follows immediately.

Case (Runs).

$$\text{(Runs)} \frac{\mu; \Xi \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi \vdash v_2 \approx v'_2 : ? \quad \mu; \Xi; z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi \vdash \text{let } \{z\}_{v_1} = v_2 \text{ in } t_3 \approx \text{let } z = \varepsilon_{\rightarrow?} \pi_2(\varepsilon_{\times?} v'_1 :: ? \times ?) :: ? \rightarrow ? v'_2 \text{ in } t'_3 : ?}$$

Since $t = \text{let } \{z\}_{v_1} = v_2 \text{ in } t_3$, we know that $t \parallel \mu \mapsto \mathbf{error}$. Therefore, by Lemma A.132, the result follows immediately. □

Lemma A.136 *If $\varepsilon = \langle E, E \rangle$, then $\varepsilon \circ \varepsilon = \varepsilon$.*

PROOF. Straightforward induction on the shape of the evidence ε . □

Lemma A.137 *If $\Xi; \Gamma \vdash [t]_\varepsilon : G$ then $G = ?$.*

PROOF. Straightforward induction on the syntax of t . □

Lemma A.138 *If $\mu; \Xi; \Gamma \vdash t \approx t_\varepsilon : ?$ then $\Xi; \Gamma \vdash t_\varepsilon : ?$.*

PROOF. Direct by Lemma A.137. □

Lemma A.139 *If t is closed λ_{seal} term, then $\cdot; \cdot; \cdot \vdash [t]_\varepsilon : ?$.*

PROOF. Straightforward induction on the syntax of t . □

Lemma A.140 $\Xi \triangleright su_\varepsilon \mapsto^* \Xi, \sigma := ? \triangleright su_\varepsilon^\sigma$, where $\sigma := ? \notin \Xi$.

PROOF. Following the reduction rules of GSF. □

Lemma A.141 (Substitution preserves) *If $\mu; \Xi; \Gamma, x : ? \vdash t \approx t^* : ?$ and $\mu; \Xi; \Gamma \vdash v \approx v^* : ?$, then $\mu; \Xi; \Gamma \vdash t[v/x] \approx t^*[v^*/x] : ?$.*

PROOF. The proof is a straightforward induction on the derivation of $\mu; \Xi; \Gamma, x : ? \vdash t \approx t^* : ?$.

Case (Rx).

$$(Rx) \frac{x : ? \in \Gamma, x : ?}{\mu; \Xi; \Gamma, x : ? \vdash x \approx x : ?}$$

We have that $t = x$ and $t^* = x$. By the definition of substitution, we have that $x[v/x] = v$ and $x[v^*/x] = v^*$. Therefore, we are required to prove that $\mu; \Xi; \Gamma \vdash v \approx v^* : ?$, which follows by the premise.

If we have

$$(Rx) \frac{y : ? \in \Gamma, x : ?}{\mu; \Xi; \Gamma, x : ? \vdash y \approx y : ?}$$

We have that $t = y$ and $t^* = y$. By the definition of substitution, we have that $y[v/x] = y$ and $y[v^*/x] = y$. Therefore, we are required to prove that $\mu; \Xi; \Gamma \vdash y \approx y : ?$, which follows by the premise $\mu; \Xi; \Gamma, x : ? \vdash y \approx y : ?$ and Lemma A.134.

Case (Rb).

$$(Rb) \frac{ty(b) = B}{\mu; \Xi; \Gamma, x : ? \vdash b \approx \varepsilon_B b :: ? : ?}$$

We have that $t = b$ and $t^* = \varepsilon_B b :: ?$. By the definition of substitution, we have that $b[v/x] = b$ and $\varepsilon_B b :: ?[v^*/x] = \varepsilon_B b :: ?$. Therefore, we are required to prove that $\mu; \Xi; \Gamma \vdash b \approx \varepsilon_B b :: ? : ?$, which follows by the premise $\mu; \Xi; \Gamma, x : ? \vdash b \approx \varepsilon_B b :: ? : ?$ and Lemma A.134.

Case (Ru).

$$(Ru) \frac{\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx \varepsilon_D u :: ? : ?}{\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx \varepsilon_D(\varepsilon_D u :: D) :: ? : ?}$$

We have that $t = v_1$ and $t^* = \varepsilon_D(\varepsilon_D u :: D) :: ?$. By the definition of substitution, we have that $(\varepsilon_D(\varepsilon_D u :: D) :: ?)[v^*/x] = \varepsilon_D(\varepsilon_D u[v^*/x] :: D) :: ?$. Therefore, we are required to prove that $\mu; \Xi; \Gamma \vdash v_1[v/x] \approx \varepsilon_D(\varepsilon_D u[v^*/x] :: D) :: ? : ?$, or what is the same $\mu; \Xi; \Gamma \vdash v_1[v/x] \approx (\varepsilon_D u[v^*/x] :: ?) : ?$ which follows by the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx \varepsilon_D u :: ? : ?$.

Case (Rs).

$$(Rs) \frac{\sigma := ? \in \Xi}{\mu; \Xi; \Gamma, x : ? \vdash \sigma \approx su_\varepsilon^\sigma : ?}$$

We have that $t = \sigma$ and $t^* = su_\varepsilon^\sigma$. By the definition of substitution, we have that $\sigma[v/x] = \sigma$ and $su_\varepsilon^\sigma[v^*/x] = su_\varepsilon^\sigma$. Therefore, we are required to prove that $\mu; \Xi; \Gamma \vdash \sigma \approx su_\varepsilon^\sigma : ?$, which follows by the premise $\mu; \Xi; \Gamma, x : ? \vdash \sigma \approx su_\varepsilon^\sigma : ?$ and Lemma A.134.

Case (Rp).

$$(Rp) \frac{\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx \varepsilon_{D_1} u_1 :: ? : ? \quad \mu; \Xi; \Gamma, x : ? \vdash v_2 \approx \varepsilon_{D_2} u_2 :: ? : ?}{\mu; \Xi; \Gamma, x : ? \vdash \langle v_1, v_2 \rangle \approx \varepsilon_{D_1 \times D_2} \langle u_1, u_2 \rangle :: ? : ?}$$

We have that $t = \langle v_1, v_2 \rangle$ and $t^* = \varepsilon_{D_1 \times D_2} \langle u_1, u_2 \rangle :: ?$. By the definition of substitution, we have that $\langle v_1, v_2 \rangle[v/x] = \langle v_1[v/x], v_2[v/x] \rangle$ and $(\varepsilon_{D_1 \times D_2} \langle u_1, u_2 \rangle :: ?)[v^*/x] = \varepsilon_{D_1 \times D_2} \langle u_1[v^*/x], u_2[v^*/x] \rangle :: ?$. Therefore, we are required to prove that $\mu; \Xi; \Gamma \vdash \langle v_1[v/x], v_2[v/x] \rangle \approx \varepsilon_{D_1 \times D_2} \langle u_1[v^*/x], u_2[v^*/x] \rangle :: ? : ?$, or what is the same by Rule (Rp) that $\mu; \Xi; \Gamma \vdash v_1[v/x] \approx \varepsilon_{D_1} u_1[v^*/x] :: ? : ?$ and $\mu; \Xi; \Gamma \vdash v_2[v/x] \approx \varepsilon_{D_2} u_2[v^*/x] :: ? : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx \varepsilon_{D_1} u_1 :: ? : ?$ and $\mu; \Xi; \Gamma, x : ? \vdash v_2 \approx \varepsilon_{D_2} u_2 :: ? : ?$ the result follows immediately.

Case (Rλ).

$$(R\lambda) \frac{\Xi; \Gamma, x : ?, y : ? \vdash t_1 \approx t_2 : ?}{\mu; \Xi; \Gamma, x : ? \vdash (\lambda y. t_1) \approx \varepsilon_{\gamma \rightarrow ?}(\lambda y. t_2) :: ? : ?}$$

We have that $t = (\lambda y. t_1)$ and $t^* = \varepsilon_{\gamma \rightarrow ?}(\lambda y. t_2) :: ?$. By the definition of substitution, we have that $(\lambda y. t_1)[v/x] = (\lambda y. t_1[v/x])$ and $(\varepsilon_{\gamma \rightarrow ?}(\lambda y. t_2) :: ?)[v^*/x] = \varepsilon_{\gamma \rightarrow ?}(\lambda y. t_2[v^*/x]) :: ?$. Therefore, we are required to prove that $\mu; \Xi; \Gamma \vdash (\lambda y. t_1[v/x]) \approx \varepsilon_{\gamma \rightarrow ?}(\lambda y. t_2[v^*/x]) :: ? : ?$, or what is the same $\mu; \Xi; \Gamma, y : ? \vdash t_1[v/x] \approx t_2[v^*/x] : ?$ which follows by the induction hypothesis on $\mu; \Xi; \Gamma, x : ?, y : ? \vdash t_1 \approx t_2 : ?$.

Case (Rpt).

$$(Rpt) \frac{\mu; \Xi; \Gamma, x : ? \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t'_2 : ?}{\mu; \Xi; \Gamma, x : ? \vdash \langle t_1, t_2 \rangle \approx \varepsilon_{\gamma \times ?} \langle t'_1, t'_2 \rangle :: ? : ?}$$

We have that $t = \langle t_1, t_2 \rangle$ and $t^* = \varepsilon_{\gamma \times ?} \langle t'_1, t'_2 \rangle :: ?$. By the definition of substitution, we have that $\langle t_1, t_2 \rangle[v/x] = \langle t_1[v/x], t_2[v/x] \rangle$ and $(\varepsilon_{\gamma \times ?} \langle t'_1, t'_2 \rangle :: ?)[v^*/x] = (\varepsilon_{\gamma \times ?} \langle t'_1[v^*/x], t'_2[v^*/x] \rangle :: ?)$. Therefore, we are required to prove that $\mu; \Xi; \Gamma \vdash \langle t_1[v/x], t_2[v/x] \rangle \approx (\varepsilon_{\gamma \times ?} \langle t'_1[v^*/x], t'_2[v^*/x] \rangle :: ?) : ?$, or what is the same by Rule (Rpt) that $\mu; \Xi; \Gamma \vdash t_1[v/x] \approx t'_1[v^*/x] : ?$ and $\mu; \Xi; \Gamma \vdash t_2[v/x] \approx t'_2[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash t_1 \approx t'_1 : ?$ and $\mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t'_2 : ?$ the result follows immediately.

Case (Rapp).

$$(Rapp) \frac{\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma, x : ? \vdash v_2 \approx v'_2 : ?}{\mu; \Xi; \Gamma, x : ? \vdash v_1 v_2 \approx (\varepsilon_{\gamma \rightarrow ?} v'_1 :: ? \rightarrow ?) v'_2 : ?}$$

We have that $t = v_1 v_2$ and $t^* = (\varepsilon_{\gamma \rightarrow ?} v'_1 :: ? \rightarrow ?) v'_2$. By the definition of substitution, we have that

$$(v_1 v_2)[v/x] = v_1[v/x] v_2[v/x]$$

and

$$((\varepsilon_{\rightarrow?} v'_1 :: ? \rightarrow ?) v'_2)[v^*/x] = (\varepsilon_{\rightarrow?} v'_1[v^*/x] :: ? \rightarrow ?) v'_2[v^*/x]$$

Therefore, we are required to prove that

$$\mu; \Xi; \Gamma \vdash v_1[v/x] v_2[v/x] \approx (\varepsilon_{\rightarrow?} v'_1[v^*/x] :: ? \rightarrow ?) v'_2[v^*/x] : ?$$

, or what is the same by Rule (Rapp) that $\mu; \Xi; \Gamma \vdash v_1[v/x] \approx v'_1[v^*/x] : ?$ and $\mu; \Xi; \Gamma \vdash v_2[v/x] \approx v'_2[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx v'_1 : ?$ and $\mu; \Xi; \Gamma, x : ? \vdash v_2 \approx v'_2 : ?$ the result follows immediately.

Case (RappL).

$$\text{(RappL)} \frac{\mu; \Xi; \Gamma, x : ? \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t'_2 : ?}{\mu; \Xi; \Gamma, x : ? \vdash t_1 t_2 \approx \text{let } z = t'_1 \text{ in let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow?} z :: ? \rightarrow ?) y : ?}$$

We have that $t = t_1 t_2$ and $t^* = \text{let } z = t'_1 \text{ in let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow?} z :: ? \rightarrow ?) y$. By the definition of substitution, we have that

$$(t_1 t_2)[v/x] = t_1[v/x] t_2[v/x]$$

and

$$(\text{let } z = t'_1 \text{ in let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow?} z :: ? \rightarrow ?) y)[v^*/x] = \text{let } z = t'_1[v^*/x] \text{ in let } y = t'_2[v^*/x] \text{ in } (\varepsilon_{\rightarrow?} z :: ? \rightarrow ?) y$$

Therefore, we are required to prove that

$$\mu; \Xi; \Gamma \vdash t_1[v/x] t_2[v/x] \approx \text{let } z = t'_1[v^*/x] \text{ in let } y = t'_2[v^*/x] \text{ in } (\varepsilon_{\rightarrow?} z :: ? \rightarrow ?) y : ?$$

, or what is the same by Rule (RappL) that $\mu; \Xi; \Gamma \vdash t_1[v/x] \approx t'_1[v^*/x] : ?$ and $\mu; \Xi; \Gamma \vdash t_2[v/x] \approx t'_2[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash t_1 \approx t'_1 : ?$ and $\mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t'_2 : ?$ the result follows immediately.

Case (RappR).

$$\text{(RappR)} \frac{\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t'_2 : ?}{\mu; \Xi; \Gamma, x : ? \vdash v_1 t_2 \approx \text{let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow?} v'_1 :: ? \rightarrow ?) y : ?}$$

We have that $t = v_1 t_2$ and $t^* = \text{let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow?} v'_1 :: ? \rightarrow ?) y$. By the definition of substitution, we have that

$$(v_1 t_2)[v/x] = v_1[v/x] t_2[v/x]$$

and

$$(\text{let } y = t'_2 \text{ in } (\varepsilon_{\rightarrow?} v'_1 :: ? \rightarrow ?) y)[v^*/x] = \text{let } y = t'_2[v^*/x] \text{ in } (\varepsilon_{\rightarrow?} v'_1[v^*/x] :: ? \rightarrow ?) y$$

Therefore, we are required to prove that

$$\mu; \Xi; \Gamma \vdash v_1[v/x] t_2[v/x] \approx \text{let } y = t'_2[v^*/x] \text{ in } (\varepsilon_{\rightarrow?} v'_1[v^*/x] :: ? \rightarrow ?) y : ?$$

, or what is the same by Rule (RappR) that $\mu; \Xi; \Gamma \vdash v_1[v/x] \approx v'_1[v^*/x] : ?$ and $\mu; \Xi; \Gamma \vdash t_2[v/x] \approx t'_2[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx v'_1 : ?$ and $\mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t'_2 : ?$ the result follows immediately.

Case (R?).

$$\text{(R?)} \frac{\mu; \Xi; \Gamma, x : ? \vdash t \approx t' : ?}{\mu; \Xi; \Gamma, x : ? \vdash t \approx \varepsilon_{\rightarrow?} t' :: ? : ?}$$

We have that $t^* = \varepsilon_{\gamma} t' :: ?$. By the definition of substitution, we have that

$$(\varepsilon_{\gamma} t' :: ?)[v^*/x] = \varepsilon_{\gamma} t'[v^*/x] :: ?$$

Therefore, we are required to prove that

$$\mu; \Xi; \Gamma \vdash t[v/x] \approx \varepsilon_{\gamma} t'[v^*/x] :: ? : ?$$

, or what is the same by Rule (R?) that $\mu; \Xi; \Gamma \vdash t[v/x] \approx t'[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash t \approx t' : ?$ the result follows immediately.

Case (Rpi).

$$\text{(Rpi)} \frac{\mu; \Xi; \Gamma, x : ? \vdash t'' \approx t' : ?}{\mu; \Xi; \Gamma, x : ? \vdash \pi_i(t'') \approx \pi_i(\varepsilon_{\gamma \times \gamma} t' :: ? \times ?) : ?}$$

We have that $t = \pi_i(t'')$ and $t^* = \pi_i(\varepsilon_{\gamma \times \gamma} t' :: ? \times ?)$. By the definition of substitution, we have that

$$\pi_i(t'')[v/x] = \pi_i(t''[v/x])$$

and

$$(\pi_i(\varepsilon_{\gamma \times \gamma} t' :: ? \times ?))[v^*/x] = \pi_i(\varepsilon_{\gamma \times \gamma} t'[v^*/x] :: ? \times ?)$$

Therefore, we are required to prove that

$$\mu; \Xi; \Gamma \vdash \pi_i(t''[v/x]) \approx \pi_i(\varepsilon_{\gamma \times \gamma} t'[v^*/x] :: ? \times ?) : ?$$

, or what is the same by Rule (Rpi) that $\mu; \Xi; \Gamma \vdash t''[v/x] \approx t'[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash t'' \approx t' : ?$ the result follows immediately.

Case (RsG).

$$\text{(RsG)} \frac{\mu; \Xi; \Gamma, x : ?, z : ? \vdash t'' \approx t' : ?}{\mu; \Xi; \Gamma, x : ? \vdash \nu z.t'' \approx \text{let } z = su_{\varepsilon} \text{ in } t' : ?}$$

We have that $t = \nu z.t''$ and $t^* = \text{let } z = su_{\varepsilon} \text{ in } t'$. By the definition of substitution, we have that

$$(\nu z.t'')[v/x] = \nu z.t''[v/x]$$

and

$$(\text{let } z = su_{\varepsilon} \text{ in } t')[v^*/x] = \text{let } z = su_{\varepsilon} \text{ in } t'[v^*/x]$$

Therefore, we are required to prove that

$$\mu; \Xi; \Gamma \vdash \nu z.t''[v/x] \approx \text{let } z = su_{\varepsilon} \text{ in } t'[v^*/x] : ?$$

, or what is the same by Rule (RsG) that $\mu; \Xi; \Gamma, z : ? \vdash t''[v/x] \approx t'[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ?, z : ? \vdash t'' \approx t' : ?$ the result follows immediately.

Case (Rsed1).

$$\text{(Rsed1)} \frac{\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma, x : ? \vdash v_2 \approx v'_2 : ?}{\mu; \Xi; \Gamma, x : ? \vdash \{v_1\}_{v_2} \approx \varepsilon_{\gamma \rightarrow \gamma} \pi_1(\varepsilon_{\gamma \times \gamma} v'_2 :: ? \times ?) :: ? \rightarrow ? v'_1 : ?}$$

We have that $t = \{v_1\}_{v_2}$ and $t^* = \varepsilon_{\gamma \rightarrow \gamma} \pi_1(\varepsilon_{\gamma \times \gamma} v'_2 :: ? \times ?) :: ? \rightarrow ? v'_1$. By the definition of substitution, we have that

$$\{v_1\}_{v_2}[v/x] = \{v_1[v/x]\}_{v_2[v/x]}$$

and

$$(\varepsilon_{\gamma \rightarrow \gamma} \pi_1(\varepsilon_{\gamma \times \gamma} v'_2 :: ? \times ?) :: ? \rightarrow ? v'_1)[v^*/x] = \varepsilon_{\gamma \rightarrow \gamma} \pi_1(\varepsilon_{\gamma \times \gamma} v'_2[v^*/x] :: ? \times ?) :: ? \rightarrow ? v'_1[v^*/x]$$

Therefore, we are required to prove that

$$\mu; \Xi; \Gamma \vdash \{v_1[v/x]\}_{v_2[v/x]} \approx \varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} v_2'[v^*/x] :: ? \times ?) :: ? \rightarrow ? v_1'[v^*/x] : ?$$

, or what is the same by Rule (Rsed1) that $\mu; \Xi; \Gamma \vdash v_1[v/x] \approx v_1'[v^*/x] : ?$ and $\mu; \Xi; \Gamma \vdash v_2[v/x] \approx v_2'[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx v_1' : ?$ and $\mu; \Xi; \Gamma, x : ? \vdash v_2 \approx v_2' : ?$ the result follows immediately.

Case (Rsed1L).

$$\text{(Rsed1L)} \frac{\mu; \Xi; \Gamma, x : ? \vdash t_1 \approx t_1' : ? \quad \mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t_2' : ?}{\mu; \Xi; \Gamma, x : ? \vdash \{t_1\}_{t_2} \approx \text{let } z = t_1' \text{ in let } y = t_2' \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) z : ?}$$

We have that $t = \{t_1\}_{t_2}$ and $t^* = \text{let } z = t_1' \text{ in let } y = t_2' \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) z$. By the definition of substitution, we have that

$$\{t_1\}_{t_2}[v/x] = \{t_1[v/x]\}_{t_2[v/x]}$$

and

$$\begin{aligned} & (\text{let } z = t_1' \text{ in let } y = t_2' \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) z)[v^*/x] = \\ & \text{let } z = t_1'[v^*/x] \text{ in let } y = t_2'[v^*/x] \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) z \end{aligned}$$

Therefore, we are required to prove that

$$\mu; \Xi; \Gamma \vdash \{t_1[v/x]\}_{t_2[v/x]} \approx \text{let } z = t_1'[v^*/x] \text{ in let } y = t_2'[v^*/x] \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) z : ?$$

, or what is the same by Rule (Rsed1L) that $\mu; \Xi; \Gamma \vdash t_1[v/x] \approx t_1'[v^*/x] : ?$ and $\mu; \Xi; \Gamma \vdash t_2[v/x] \approx t_2'[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash t_1 \approx t_1' : ?$ and $\mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t_2' : ?$ the result follows immediately.

Case (Rsed1R).

$$\text{(Rsed1R)} \frac{\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx v_1' : ? \quad \mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t_2' : ?}{\mu; \Xi; \Gamma, x : ? \vdash \{v_1\}_{t_2} \approx \text{let } y = t_2' \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) v_1' : ?}$$

We have that $t = \{v_1\}_{t_2}$ and $t^* = \text{let } y = t_2' \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) v_1'$. By the definition of substitution, we have that

$$\{v_1\}_{t_2}[v/x] = \{v_1[v/x]\}_{t_2[v/x]}$$

and

$$\begin{aligned} & (\text{let } y = t_2' \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) v_1')[v^*/x] = \\ & \text{let } y = t_2'[v^*/x] \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) v_1'[v^*/x] \end{aligned}$$

Therefore, we are required to prove that

$$\mu; \Xi; \Gamma \vdash \{v_1[v/x]\}_{t_2[v/x]} \approx \text{let } y = t_2'[v^*/x] \text{ in } (\varepsilon_{\rightarrow?} \pi_1(\varepsilon_{\times?} y :: ? \times ?) :: ? \rightarrow ?) v_1'[v^*/x] : ?$$

, or what is the same by Rule (Rsed1R) that $\mu; \Xi; \Gamma \vdash v_1[v/x] \approx v_1'[v^*/x] : ?$ and $\mu; \Xi; \Gamma \vdash t_2[v/x] \approx t_2'[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx v_1' : ?$ and $\mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t_2' : ?$ the result follows immediately.

Case (Rsed2).

$$\text{(Rsed2)} \frac{\mu; \Xi; \Gamma, x : ? \vdash v' \approx \langle E_1, E_2 \rangle u :: ? : ? \quad \sigma := ? \in \Xi}{\mu; \Xi; \Gamma, x : ? \vdash \{v'\}_\sigma \approx \langle E_1, \sigma^{E_2} \rangle u :: ? : ?}$$

We have that $t = \{v'\}_\sigma$ and $t^* = \langle E_1, \sigma^{E_2} \rangle u :: ?$. By the definition of substitution, we have that

$$\{v'\}_\sigma[v/x] = \{v'[v/x]\}_\sigma$$

and

$$\langle \langle E_1, \sigma^{E_2} \rangle u :: ? \rangle [v^*/x] = \langle \langle E_1, \sigma^{E_2} \rangle u [v^*/x] :: ? \rangle$$

Therefore, we are required to prove that

$$\mu; \Xi; \Gamma \vdash \{v'[v/x]\}_\sigma \approx \langle \langle E_1, \sigma^{E_2} \rangle u [v^*/x] :: ? \rangle : ?$$

, or what is the same by Rule (Rsd2) that $\mu; \Xi; \Gamma \vdash v'[v/x] \approx \langle E_1, E_2 \rangle u [v^*/x] :: ? : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash v' \approx \langle E_1, E_2 \rangle u :: ? : ?$ the result follows immediately.

Case (Runs).

$$\text{(Runs)} \frac{\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma, x : ? \vdash v_2 \approx v'_2 : ? \quad \mu; \Xi; \Gamma, x : ?, z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi; \Gamma, x : ? \vdash \text{let } \{z\}_{v_1} = v_2 \text{ in } t_3 \approx \text{let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1 :: ? \times ?) :: ? \rightarrow ? v'_2 \text{ in } t'_3 : ?}$$

We have that $t = \text{let } \{z\}_{v_1} = v_2 \text{ in } t_3$ and $t^* = \text{let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1 :: ? \times ?) :: ? \rightarrow ? v'_2 \text{ in } t'_3$. By the definition of substitution, we have that

$$(\text{let } \{z\}_{v_1} = v_2 \text{ in } t_3)[v/x] = \text{let } \{z\}_{v_1[v/x]} = v_2[v/x] \text{ in } t_3[v/x]$$

and

$$\begin{aligned} & (\text{let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1 :: ? \times ?) :: ? \rightarrow ? v'_2 \text{ in } t'_3)[v^*/x] = \\ & \text{let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1[v^*/x] :: ? \times ?) :: ? \rightarrow ? v'_2[v^*/x] \text{ in } t'_3[v^*/x] \end{aligned}$$

Therefore, we are required to prove that

$$\mu; \Xi; \Gamma \vdash \text{let } \{z\}_{v_1[v/x]} = v_2[v/x] \text{ in } t_3[v/x] \approx \text{let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1[v^*/x] :: ? \times ?) :: ? \rightarrow ? v'_2[v^*/x] \text{ in } t'_3[v^*/x] : ?$$

Or what is the same by Rule (Runs) that $\mu; \Xi; \Gamma \vdash v_1[v/x] \approx v'_1[v^*/x] : ?$, $\mu; \Xi; \Gamma \vdash v_2[v/x] \approx v'_2[v^*/x] : ?$ and $\mu; \Xi; \Gamma, z : ? \vdash t_3[v^*/x] \approx t'_3[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx v'_1 : ?$, $\mu; \Xi; \Gamma, x : ? \vdash v_2 \approx v'_2 : ?$ and $\mu; \Xi; \Gamma, x : ?, z : ? \vdash t_3 \approx t'_3 : ?$ the result follows immediately.

Case (RunsL).

$$\text{(RunsL)} \frac{\mu; \Xi; \Gamma, x : ? \vdash t_1 \approx t'_1 : ? \quad \mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t'_2 : ? \quad \mu; \Xi; \Gamma, x : ?, z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi; \Gamma, x : ? \vdash \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3 \approx \text{let } w = t'_1 \text{ in let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} w :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3 : ?}$$

We have that $t = \text{let } \{z\}_{t_1} = t_2 \text{ in } t_3$ and

$$t^* = \text{let } w = t'_1 \text{ in let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} w :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3$$

By the definition of substitution, we have that

$$(\text{let } \{z\}_{t_1} = t_2 \text{ in } t_3)[v/x] = \text{let } \{z\}_{t_1[v/x]} = t_2[v/x] \text{ in } t_3[v/x]$$

and

$$\begin{aligned} & (\text{let } w = t'_1 \text{ in let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} w :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3)[v^*/x] = \\ & \text{let } w = t'_1[v^*/x] \text{ in let } y = t'_2[v^*/x] \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} w :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3[v^*/x] \end{aligned}$$

Therefore, we are required to prove that

$$\Xi; \Gamma \vdash \text{let } \{z\}_{t_1[v/x]} = t_2[v/x] \text{ in } t_3[v/x] \approx$$

$$\text{let } w = t'_1[v^*/x] \text{ in let } y = t'_2[v^*/x] \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} w :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3[v^*/x] : ?$$

Or what is the same by Rule (RunsL) that $\mu; \Xi; \Gamma \vdash t_1[v/x] \approx t'_1[v^*/x] : ?$, $\mu; \Xi; \Gamma \vdash t_2[v/x] \approx t'_2[v^*/x] : ?$ and $\mu; \Xi; \Gamma, z : ? \vdash t_3[v^*/x] \approx t'_3[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash t_1 \approx t'_1 : ?$, $\mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t'_2 : ?$ and $\mu; \Xi; \Gamma, x : ?, z : ? \vdash t_3 \approx t'_3 : ?$ the result follows immediately.

Case (RunsR).

$$\text{(RunsR)} \frac{\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx v'_1 : ? \quad \mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t'_2 : ? \quad \mu; \Xi; \Gamma, x : ?, z : ? \vdash t_3 \approx t'_3 : ?}{\mu; \Xi; \Gamma, x : ? \vdash \text{let } \{z\}_{v_1} = t_2 \text{ in } t_3 \approx \text{let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1 :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3 : ?}$$

We have that $t = \text{let } \{z\}_{v_1} = t_2 \text{ in } t_3$ and

$$t^* = \text{let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1 :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3$$

By the definition of substitution, we have that

$$(\text{let } \{z\}_{v_1} = t_2 \text{ in } t_3)[v/x] = \text{let } \{z\}_{v_1[v/x]} = t_2[v/x] \text{ in } t_3[v/x]$$

and

$$\begin{aligned} & (\text{let } y = t'_2 \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1 :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3)[v^*/x] = \\ & \text{let } y = t'_2[v^*/x] \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1[v^*/x] :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3[v^*/x] \end{aligned}$$

Therefore, we are required to prove that

$$\begin{aligned} & \Xi; \Gamma \vdash \text{let } \{z\}_{v_1[v/x]} = t_2[v/x] \text{ in } t_3[v/x] \approx \\ & \text{let } y = t'_2[v^*/x] \text{ in let } z = \varepsilon_{? \rightarrow ?} \pi_2(\varepsilon_{? \times ?} v'_1[v^*/x] :: ? \times ?) :: ? \rightarrow ? y \text{ in } t'_3[v^*/x] : ? \end{aligned}$$

Or what is the same by Rule (RunsR) that $\mu; \Xi; \Gamma \vdash v_1[v/x] \approx v'_1[v^*/x] : ?$, $\mu; \Xi; \Gamma \vdash t_2[v/x] \approx t'_2[v^*/x] : ?$ and $\mu; \Xi; \Gamma, z : ? \vdash t_3[v^*/x] \approx t'_3[v^*/x] : ?$. By the induction hypothesis on $\mu; \Xi; \Gamma, x : ? \vdash v_1 \approx v'_1 : ?$, $\mu; \Xi; \Gamma, x : ? \vdash t_2 \approx t'_2 : ?$ and $\mu; \Xi; \Gamma, x : ?, z : ? \vdash t_3 \approx t'_3 : ?$ the result follows immediately.

□

Lemma A.142 *If $\mu; \Xi; \Gamma, x : ? \vdash t \approx t_\varepsilon : ?$ and $\mu; \Xi; \Gamma \vdash v \approx v_\varepsilon : ?$, then $\mu; \Xi; \Gamma \vdash t[v/x] \approx t_\varepsilon[v_\varepsilon/x] : ?$.*

PROOF. Direct by Lemma A.141.

□

The remaining theorems and lemmas are in the main text.

A.10 Gradual Existential Types in GSF

This session presents a motivational example for the extension of GSF with existential directly instead of using the encoding of existential into universal types. Also, we show the translation from GSF^\exists to $\text{GSF}_\varepsilon^\exists$ and the proof of the fundamental property for existential types.

A.10.1 Existential types: primitive or encoded?

The benefit of a direct treatment of existential types can already be appreciated in the fully-static setting, with the simple examples of packages s_1 and s_2 above. Suppose we want to show that s_1 and s_2 are contextually equivalent, *i.e.* indistinguishable by any context. To show this equivalence, it is sufficient to show that the packages are logically related. The proof of this based on the direct interpretation of the existential types is considerably easier and more intuitive than proving that their *encodings* are related. To illustrate this point, we sketch these two proof techniques below in System F.

Proof with primitive existentials. Two packages are logically related at an existential type, if there exists a relation R between values of their representation types, such that their term components respect the relation R . Here, for v_1 and v_2 to respect R means that the following three conditions hold:

- The created semaphores with the operation *bit* are related. In this case, this imposes that $(\text{true}, 1) \in R$.
- If two semaphores are related, then changing their states with the operation *flip* yields related semaphores. Here, applying the *flip* operation of each package s_1 and s_2 to the values *true* and 1, respectively, yields *false* and 0. Therefore, $(\text{false}, 0) \in R$. Applying the *flip* operations on these values yields again *true* and 1, which are related.
- If two semaphores are related, then the *Bool* value obtained by applying the operation *read* must be the same. It is easy to see that this condition is also satisfied.

Formally, two packages are logically related at an existential type in standard System F (following (Ahmed, 2006)):

$$\mathcal{V}_\rho \llbracket \exists X.T \rrbracket = \{ (\text{pack}\langle T_1, v_1 \rangle \text{ as } \exists X.\rho(T), \text{pack}\langle T_2, v_2 \rangle \text{ as } \exists X.\rho(T)) \in \text{Atom}_\rho^\exists[\exists X.T] \mid \exists R \in \text{REL}[T_1, T_2].(v_1, v_2) \in \mathcal{V}_{\rho[X \mapsto (R, T_1, T_2)]} \llbracket T \rrbracket \}$$

By this definition, in order to prove that s_1 is logically related to s_2 at type *Sem*, it is required to show that there exists a relation R between the types *Bool* and *Int* such that

$$(v_1, v_2) \in \mathcal{V}_{[X \mapsto (R, \text{Bool}, \text{Int})]} \llbracket X \times (X \rightarrow X) \times (X \rightarrow \text{Bool}) \rrbracket$$

Taking $R = \{ \langle \text{true}, 1 \rangle, \langle \text{false}, 0 \rangle \}$, it is easy to check that the implementations of s_1 and s_2 preserve this relation.

Proof with encoded existentials. Using the encoding of *Sem* in terms of universal types in order to prove that s_1 and s_2 are logically related is considerably more complex. First, we have to

transform the packages s_1 and s_2 to type abstractions and prove that

$$((\Lambda Y. \lambda f : Sem_{client}.f [\mathbf{Bool}] v_1), (\Lambda Y. \lambda f : Sem_{client}.f [\mathbf{Int}] v_2)) \in \mathcal{V}_\rho \llbracket \forall Y. Sem_{client} \rightarrow Y \rrbracket$$

where $Sem_{client} = \forall X. X \times (X \rightarrow X) \times (X \rightarrow \mathbf{Bool}) \rightarrow Y$. The proof of the above leads us to show that for any type T'_1 and T'_2 , and any relation R' between these types, the following type applications are related:

$$((\Lambda Y. \lambda f : Sem_{client}.f [\mathbf{Bool}] v_1) [T'_1], (\Lambda Y. \lambda f : Sem_{client}.f [\mathbf{Int}] v_2) [T'_2]) \in \mathcal{J}_{[Y \mapsto (R', T'_1, T'_2)]} \llbracket Sem_{client} \rightarrow Y \rrbracket$$

Several steps further in the proof, we have to show that $(f_1 [\mathbf{Bool}] v_1, f_2 [\mathbf{Int}] v_2) \in \mathcal{J}_{[Y \mapsto (R', T'_1, T'_2)]} \llbracket Y \rrbracket$, for any f_1 and f_2 such that

$$(f_1, f_2) \in \mathcal{V}_{[Y \mapsto (R', T'_1, T'_2)]} \llbracket Sem_{client} \rrbracket$$

Since f_1 and f_2 are related under a universal type, we can instantiate them at any types T_1 and T_2 , and any relation Q between these types, keeping the resulting terms related:

$$(f_1 [T_1], f_2 [T_2]) \in \mathcal{J}_{[Y \mapsto (R', T'_1, T'_2), X \mapsto (Q, T_1, T_2)]} \llbracket (X \times (X \rightarrow X) \times (X \rightarrow \mathbf{Bool}) \rightarrow Y) \rrbracket$$

At this point, we can pick the same relation as above, $R = \{\langle \mathbf{true}, 1 \rangle, \langle \mathbf{false}, 0 \rangle\}$, such that v_1 and v_2 are related.

$$(v_1, v_2) \in \mathcal{V}_{[X \mapsto (R, \mathbf{Bool}, \mathbf{Int})]} \llbracket X \times (X \rightarrow X) \times (X \rightarrow \mathbf{Bool}) \rrbracket$$

Hence, we can instantiate T_1 and T_2 with the types \mathbf{Bool} and \mathbf{Int} , and Q with the relation R , obtaining that

$$(f_1 [\mathbf{Bool}], f_2 [\mathbf{Int}]) \in \mathcal{J}_{[Y \mapsto (R', T'_1, T'_2), X \mapsto (R, \mathbf{Bool}, \mathbf{Int})]} \llbracket (X \times (X \rightarrow X) \times (X \rightarrow \mathbf{Bool}) \rightarrow Y) \rrbracket$$

In a few more steps, we can instantiate the above with v_1 and v_2 , since they are related, finally obtaining the desired result.

As we can see, as part of the second approach (using the encoding) is needed to prove the same that is required by the first approach (directly on existential types) and more; being the second extremely more complex. The equivalence example that we use to illustrate the previous is very simple. But, for instance, Ahmed et al. (2009a) prove challenging cases of equivalences in the presence of abstract data types and mutable references, where the use of the encoding would have hindered the work.

A.10.2 Translation from \mathbf{GSF}^\exists to $\mathbf{GSF}_\varepsilon^\exists$

Figure A.7 shows the translation from \mathbf{GSF}^\exists to $\mathbf{GSF}_\varepsilon^\exists$.

A.10.3 Properties of \mathbf{GSF}^\exists

Proposition A.143 (\mathbf{GSF}^\exists : Precision, inductively) *The inductive definition of type precision given in Figure 5.2 is equivalent to Definition 3.1.*

PROOF. Direct by induction on the type structure of G_1 and G_2 . Similar to Prop. 3.2. \square

$$\begin{array}{c}
\dots \\
(\text{Gpacku}) \frac{\Delta; \Gamma \vdash v :: G[G'/X] \rightsquigarrow v' : G[G'/X] \quad \Delta \vdash G'}{\Delta; \Gamma \vdash \text{pack}\langle G', v \rangle \text{ as } \exists X.G \rightsquigarrow \text{packu}\langle G', v' \rangle \text{ as } \exists X.G : \exists X.G} \\
(\text{Gpack}) \frac{t \not\rightsquigarrow v \quad \Delta; \Gamma \vdash t \rightsquigarrow t' : G_1 \quad \varepsilon = \mathcal{I}(\cdot, \cdot)G_1, G[G'/X] \quad \Delta \vdash G'}{\Delta; \Gamma \vdash \text{pack}\langle G', t \rangle \text{ as } \exists X.G \rightsquigarrow \text{pack}\langle G', \varepsilon t :: G[G'/X] \rangle \text{ as } \exists X.G : \exists X.G} \\
(\text{Gunpack}) \frac{\Delta; \Gamma \vdash t_1 \rightsquigarrow t'_1 : G_1 \quad G_1 \rightarrow \exists X.G'_1 \quad \varepsilon = \mathcal{I}(\cdot, \cdot)G_1, \exists X.G'_1}{\Delta, X; \Gamma, x : G'_1 \vdash t_2 \rightsquigarrow t'_2 : G_2 \quad \Delta \vdash G_2} \\
\Delta; \Gamma \vdash \text{unpack}\langle X, x \rangle = t_1 \text{ in } t_2 \rightsquigarrow \text{unpack}\langle X, x \rangle = \varepsilon t'_1 :: \exists X.G'_1 \text{ in } t'_2 : G_2
\end{array}$$

Figure A.7: Translation from GSF^\exists to $\text{GSF}_\varepsilon^\exists$

Proposition A.144 (GSF^\exists : Consistency, inductively) *The inductive definition of type consistency given in Figure 5.2 is equivalent to Definition B.96.*

PROOF. Similar to Prop. B.6.1. □

Proposition A.145 (GSF^\exists : Static equivalence for static terms) *Let t be a static term and G a static type ($G = T$). We have $\vdash_S t : T$ if and only if $\vdash t : T$.*

PROOF. Similar to Prop. 3.10. □

Proposition A.146 (GSF^\exists : Static gradual guarantee) *Let t and t' be closed GSF^\exists terms such that $t \sqsubseteq t'$ and $\vdash t : G$. Then $\vdash t' : G'$ and $G \sqsubseteq G'$.*

PROOF. Similar to Prop. 3.11. □

A.10.4 GSF^\exists : Parametricity

Theorem A.147 (Fundamental Property) *If $\Xi; \Delta; \Gamma \vdash t : G$ then $\Xi; \Delta; \Gamma \vdash t \preceq t : G$.*

We follow by induction on the structure of t .

PROOF.

Case (packu). Then $t = \varepsilon(\text{packu}\langle G', v \rangle \text{ as } \exists X.G'') :: G$, and therefore by the typing rules Epacku and Easc we have that

$$(\text{Epack \& Easc}) \frac{\Xi; \Delta; \Gamma \vdash v : G''[G'/X] \quad \Xi; \Delta \vdash G' \quad \varepsilon \Vdash \Xi; \Delta \vdash \exists X.G'' \sim G}{\Xi; \Delta; \Gamma \vdash \varepsilon(\text{packu}\langle G', v \rangle \text{ as } \exists X.G'') :: G : G}$$

Then we have to prove that:

$$\Xi; \Delta; \Gamma \vdash \varepsilon(\text{packu}\langle G', v \rangle \text{ as } \exists X.G'') :: G \preceq \varepsilon(\text{packu}\langle G', v \rangle \text{ as } \exists X.G'') :: G : G$$

By induction hypotheses we already know that $\Xi; \Delta; \Gamma \vdash v \preceq v : G''[G'/X]$. But the result follows directly by Prop A.149 (Compatibility of packu).

Case (pack). Then $t = \text{pack}\langle G', t' \rangle$ as $\exists X.G''$, and therefore by the typing rules Epack we have that

$$\text{(Epack)} \frac{\Xi; \Delta; \Gamma \vdash t' : G''[G'/X] \quad \Xi; \Delta \vdash G'}{\Xi; \Delta; \Gamma \vdash \text{pack}\langle G', t' \rangle \text{ as } \exists X.G'' : \exists X.G''}$$

Then we have to prove that:

$$\Xi; \Delta; \Gamma \vdash \text{pack}\langle G', t' \rangle \text{ as } \exists X.G'' \preceq \text{pack}\langle G', t' \rangle \text{ as } \exists X.G'' : \exists X.G''$$

By induction hypotheses we already know that $\Xi; \Delta; \Gamma \vdash t' \preceq t' : G''[G'/X]$. But the result follows directly by Prop A.150 (Compatibility of pack).

Case (unpack). Then $t = \text{unpack}\langle X, x \rangle = t_1$ in t_2 , and therefore:

$$\text{(Eunpack)} \frac{\Xi; \Delta; \Gamma \vdash t_1 : \exists X.G_1 \quad \Xi; \Delta, X; \Gamma, x : G_1 \vdash t_2 : G_2 \quad \Xi; \Delta \vdash G_2}{\Xi; \Delta; \Gamma \vdash \text{unpack}\langle X, x \rangle = t_1 \text{ in } t_2 : G_2}$$

where $G = G_2$. Then we have to prove that:

$$\Xi; \Delta; \Gamma \vdash \text{unpack}\langle X, x \rangle = t_1 \text{ in } t_2 \preceq \text{unpack}\langle X, x \rangle = t_1 \text{ in } t_2 : G_2$$

By induction hypotheses we already know that $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_1 : \exists X.G_1$ and $\Xi; \Delta, X; \Gamma, x : G_1 \vdash t_2 \preceq t_2 : G_2$. But the result follows directly by Prop A.151 (Compatibility of unpack).

□

Definition A.148 (Operators over evidence)

$$\begin{aligned} \pi_i^*(\varepsilon) &\triangleq \langle E_*, E_* \rangle \quad \text{where } E_* = \text{lift}_{\Xi}(\text{unlift}(\pi_i(\varepsilon))) & \pi_i^2(\varepsilon) &\triangleq \langle E_*, E_* \rangle \quad \text{where } E_* = \pi_i(\varepsilon) \\ \langle E_1, E_2 \rangle[X] &= \langle E_1[X], E_2[X] \rangle & \langle E_1, E_2 \rangle[E_3, E_4] &= \langle E_1[E_3], E_2[E_4] \rangle \\ \langle E_1, E_2 \rangle[E_3, E_4, X] &= \langle E_1[E_3/X], E_2[E_4/X] \rangle \end{aligned}$$

Proposition A.149 (Compatibility-Epacku) *If $\Xi; \Delta; \Gamma \vdash v_{11} \preceq v_{12} : G''[G'/X]$, $\Xi; \Delta \vdash G'$ and $\varepsilon \Vdash \Xi; \Delta \vdash \exists X.G'' \sim G$, then*

$$\Xi; \Delta; \Gamma \vdash \varepsilon(\text{packu}\langle G', v_{11} \rangle \text{ as } \exists X.G'') :: G \preceq \varepsilon(\text{packu}\langle G', v_{12} \rangle \text{ as } \exists X.G'') :: G : G$$

PROOF. First, we are required to prove that

$$\Xi; \Delta; \Gamma \vdash \varepsilon(\text{packu}\langle G', v_{1i} \rangle \text{ as } \exists X.G'') :: G : G$$

But by unfolding the premises we know that $\Xi; \Delta; \Gamma \vdash v_{1i} : G''[G'/X]$, therefore:

$$\text{(Epack \& Easc)} \frac{\Xi; \Delta; \Gamma \vdash v_{1i} : G''[G'/X] \quad \Xi; \Delta \vdash G' \quad \varepsilon \Vdash \Xi; \Delta \vdash \exists X.G'' \sim G}{\Xi; \Delta; \Gamma \vdash \varepsilon(\text{pack}\langle G', v_{1i} \rangle \text{ as } \exists X.G'') :: G : G}$$

Consider arbitrary W, ρ, γ such that $W \in \mathcal{S}[\Xi]$, $(W, \rho) \in \mathcal{D}[\Delta]$ and $(W, \gamma) \in \mathcal{G}_{\rho}[\Gamma]$. We are required to show that

$$(W, \rho(\gamma_1(\varepsilon(\text{packu}\langle G', v_{11} \rangle \text{ as } \exists X.G'') :: G)), \rho(\gamma_2(\varepsilon(\text{packu}\langle G', v_{12} \rangle \text{ as } \exists X.G'') :: G))) \in \mathcal{J}_{\rho}[G]$$

First we have to prove that:

$$W.\Xi_i \vdash \rho(\gamma_i(\varepsilon(\text{packu}\langle G', v_{1i} \rangle \text{ as } \exists X.G'') :: G)) : \rho(G)$$

As we know that $\Xi; \Delta; \Gamma \vdash \varepsilon(\text{packu}\langle G', v_{1i} \rangle \text{ as } \exists X.G'') :: G : G$, by Lemma A.110 the result follows immediately.

By definition of substitutions

$$\rho(\gamma_i(\varepsilon(\text{packu}\langle G', v_{1i} \rangle \text{ as } \exists X.G'') :: G)) = \varepsilon_i^{\rho}(\text{packu}\langle \rho(G'), \rho(\gamma_i(v_{1i})) \rangle \text{ as } \exists X.\rho(G'')) :: \rho(G)$$

where $\varepsilon_i^{\rho} = \rho_i(\varepsilon)$ and $\varepsilon_i^{\rho}.n = k$. Therefore we have to prove that

$$(W, \varepsilon_1^{\rho}(\text{packu}\langle \rho(G'), \rho(\gamma_1(v_{11})) \rangle \text{ as } \exists X.\rho(G'')) :: \rho(G), \varepsilon_2^{\rho}(\text{packu}\langle \rho(G'), \rho(\gamma_2(v_{12})) \rangle \text{ as } \exists X.\rho(G'')) :: \rho(G)) \in \mathcal{J}_{\rho}[[G]]$$

Or what is the same

$$(W, \varepsilon_1^{\rho}(\text{packu}\langle \rho(G'), \rho(\gamma_1(v_{11})) \rangle \text{ as } \exists X.\rho(G'')) :: \rho(G), \varepsilon_2^{\rho}(\text{packu}\langle \rho(G'), \rho(\gamma_2(v_{12})) \rangle \text{ as } \exists X.\rho(G'')) :: \rho(G)) \in \mathcal{V}_{\rho}[[G]]$$

The type G can be $\exists X.G'_1$, for some G'_1 , ? or a `TYPE NAME`.

Let $u_i = \text{packu}\langle \rho(G'), v_{1i} \rangle \text{ as } \exists X.\rho(G'')$ and $G^* = \exists X.G''$, we have to prove that:

$$(W', \varepsilon_1^{\rho}u_1 :: \rho(G), \varepsilon_2^{\rho}u_2 :: \rho(G)) \in \mathcal{V}_{\rho}[[G]]$$

1. If $G = \exists X.G'_1$, by the definition of $\mathcal{V}_{\rho}[[\exists X.G'_1]]$, we have to prove that $\forall W'' \succeq W, \alpha. \exists R \in \text{REL}_{W''.j}[\rho(G'), \rho(G'')]$ such that $\forall \varepsilon' \Vdash \Xi; \text{dom}(\rho) \vdash \exists X.G'_1 \sim \exists X.G'_1 (\varepsilon'.n = l)$ it is true that

$$(W^*, (\rho_1(\varepsilon) \circledast \rho_1(\varepsilon'))[\hat{G}', \hat{\alpha}]v_{11} :: \rho(G'_1)[\alpha/X], (\rho_2(\varepsilon) \circledast \rho_2(\varepsilon'))[\hat{G}_2, \hat{\alpha}]v_{12} :: \rho(G'_1)[\alpha/X]) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[[G'_1]]$$

where $W^* = W'' \boxtimes (\alpha, \rho(G'), \rho(G''), R)$.

or what is the same, we have to prove that

$$(W^*, (\rho_1(\varepsilon \circledast \varepsilon'))[\hat{G}', \hat{\alpha}]v_{11} :: \rho(G'_1)[\alpha/X], (\rho_2(\varepsilon \circledast \varepsilon'))[\hat{G}_2, \hat{\alpha}]v_{12} :: \rho(G'_1)[\alpha/X]) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[[G'_1]]$$

By Proposition A.155 (decomposition of the evidence) we know that

$$\rho_i(\varepsilon \circledast \varepsilon')[\hat{G}', \hat{\alpha}] = \pi_1^*(\rho_i(\varepsilon \circledast \varepsilon'))[\hat{G}', \hat{\alpha}] \circledast \rho_i(\varepsilon \circledast \varepsilon')[\hat{\alpha}, \hat{\alpha}]$$

Lets take $R = \mathcal{V}_{\rho}[[G']]$.

Note that

- $W^* = W'' \boxtimes (\alpha, \rho(G'), \rho(G''), \mathcal{V}_{\rho}[[G']]) \succeq W'$
- $E'_i = \text{lift}_{W^*, \Xi_i}(\rho(G'))$,
- $E_{i*} = \text{lift}_{W^*, \Xi_i}(G_{pi}), G_{pi} = \text{unlift}(\pi_1(\rho_i(\varepsilon \circledast \varepsilon')))) \sqsubseteq \rho(G'')$,

- $\rho' = \rho[X \mapsto \alpha]$,
- $\varepsilon_i^{-1} = \pi_1^*(\rho_i(\varepsilon \circ \varepsilon'))[\hat{G}', \hat{\alpha}] = \langle E_{i*}[E'_i/X], E_{i*}[\alpha^{E_i}/X] \rangle$, such that $\varepsilon_i^{-1} \Vdash W^*. \Xi_i \vdash \rho(G''[G'/X]) \sim \rho(G''[\alpha/X])$, $\alpha^{E_i} = \text{lift}_{W^*. \Xi_i}(\alpha)$, and $E'_i = \text{lift}_{W^*. \Xi_i}(\rho(G'))$, $\varepsilon_i^{-1}.n = k$ and
- $(W', v_{11}, v_{12}) \in \mathcal{V}_\rho[G''[G'/X]]$, then $(W^*, v_{11}, v_{12}) \in \mathcal{V}_\rho[G''[G'/X]]$.

By the Lemma A.153 (compositionality) we know that

$$(W^*, \pi_1(\rho_1(\varepsilon \circ \varepsilon'))[\hat{G}', \hat{\alpha}]v_{11} :: \rho'(G''), \pi_1(\rho_2(\varepsilon \circ \varepsilon'))[\hat{G}', \hat{\alpha}]v_{12} :: \rho'(G'')) \in \mathcal{F}_{\rho'}[G'']$$

or what is the same

$$(W^*, \pi_1^*(\rho_1(\varepsilon \circ \varepsilon'))[\hat{G}', \hat{\alpha}]v_{11} :: \rho(G'')[\alpha/X], \pi_1^*(\rho_2(\varepsilon \circ \varepsilon'))[\hat{G}', \hat{\alpha}]v_{12} :: \rho(G'')[\alpha/X]) \in \mathcal{F}_{\rho[X \mapsto \alpha]}[G'']$$

Then we know that

$$(\downarrow_k W^*, \varepsilon'_1 u'_1 :: \rho(G'')[\alpha/X], \varepsilon'_2 u'_2 :: \rho(G'')[\alpha/X]) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[G'']$$

where $v_{1i} = \varepsilon'_{1i} u_i :: \rho(G''[G'/X])$ and $\varepsilon'_i = \varepsilon'_{1i} \circ \pi_1^*(\rho_i(\varepsilon \circ \varepsilon'))[\hat{G}', \hat{\alpha}]$.

Note now that

- $(\downarrow_k W^*, \varepsilon'_1 u'_1 :: \rho(G'')[\alpha/X], \varepsilon'_2 u'_2 :: \rho(G'')[\alpha/X]) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[G'']$,
- $(\varepsilon \circ \varepsilon')[X] \Vdash \Xi; \Delta, X \vdash G'' \sim G'_1, (\varepsilon \circ \varepsilon')[X].n = l$
- $\downarrow_k W^* \in \mathcal{S}[\Xi]$ and $(\downarrow_k W^*, \rho') \in \mathcal{D}[\Delta, X]$,

Then, by Lemma A.152 (Ascription Lemma), we know that

$$(\downarrow_{k+l} W^*, (\varepsilon'_1 \circ \rho'_1((\varepsilon \circ \varepsilon')[X]))u'_1 :: \rho'(G'_1), (\varepsilon'_2 \circ \rho'_2((\varepsilon \circ \varepsilon')[X]))u'_2 :: \rho'(G'_1)) \in \mathcal{V}_{\rho'}[G'_1]$$

or what is the same

$$(\downarrow_{k+l} W^*, (\varepsilon'_1 \circ \rho_1(\varepsilon \circ \varepsilon'))[\hat{\alpha}, \hat{\alpha}]u'_1 :: \rho(G'_1)[\alpha/X], (\varepsilon'_2 \circ \rho_2(\varepsilon \circ \varepsilon'))[\hat{\alpha}, \hat{\alpha}]u'_2 :: \rho(G'_1)[\alpha/X]) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[G'_1]$$

The result follows immediately.

$$(W^*, (\rho_1(\varepsilon \circ \varepsilon'))[\hat{G}', \hat{\alpha}]v_{11} :: \rho(G'_1)[\alpha/X], (\rho_2(\varepsilon \circ \varepsilon'))[\hat{G}_2, \hat{\alpha}]v_{12} :: \rho(G'_1)[\alpha/X]) \in \mathcal{F}_{\rho[X \mapsto \alpha]}[G'_1]$$

2. If $G \in \text{TYPE NAME}$ then $\varepsilon = \langle H_3, \alpha^{E_4} \rangle$. Notice that as α^{E_4} cannot have free type variables therefore H_3 neither. Then $\varepsilon = \rho_i(\varepsilon)$. As α is sync, then let us call $G''' = W.\Xi_i(\alpha)$. We have to prove that:

$$(W, \langle H_3, \alpha^{E_4} \rangle u_1 :: \alpha, \langle H_3, \alpha^{E_4} \rangle u_2 :: \alpha) \in \mathcal{V}_\rho[\alpha]$$

which, by definition of $\mathcal{V}_\rho[\alpha]$, is equivalent to prove that:

$$(\downarrow W, \langle H_3, E_4 \rangle u_1 :: G''', \langle E_3, E_4 \rangle u_2 :: G''') \in \mathcal{V}_\rho[G''']$$

Then we proceed by case analysis on ε :

- (Case $\varepsilon = \langle H_3, \alpha^{\beta^{E_4}} \rangle$). We know that $\langle H_3, \alpha^{\beta^{E_4}} \rangle \vdash \Xi; \Delta \vdash G^* \sim \alpha$, then by Lemma A.114, $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash G^* \sim G'''$. As $\beta^{E_4} \sqsubseteq G'''$, then G''' can either be $?$ or β .

If $G''' = ?$, then by definition of $\mathcal{V}_\rho[[?]]$, we have to prove that the resulting values belong to $\mathcal{V}_\rho[[\beta]]$. Also as $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash G^* \sim ?$, by Lemma A.112, $\langle H_3, \beta^{E_4} \rangle \vdash \Xi; \Delta \vdash G^* \sim \beta$, and then we proceed just like this case once again (this is process is finite as there are no circular references by construction and it ends up in something different to a type name). If $G''' = \beta$ we use an analogous argument as for $G'' = ?$.

- (Case $\varepsilon = \langle H_3, \alpha^{H_4} \rangle$). We have to prove that

$$(\downarrow W, \langle H_3, H_4 \rangle u_1 :: G''', \langle H_3, H_4 \rangle u_2 :: G''') \in \mathcal{V}_\rho[[G''']]$$

By Lemma A.114, $\langle H_3, H_4 \rangle \vdash \Xi; \Delta \vdash G^* \sim G''$. Then if $G'' = ?$, we proceed as the case $G' = ?$, with the evidence $\varepsilon = \langle H_3, H_4 \rangle$. If $G'' \in \text{HEADTYPE}$, we proceed as the previous case where $G' = \forall X.G$, and the evidence $\varepsilon = \langle H_3, H_4 \rangle$.

Also, we have to prove that $(\forall \Xi', \varepsilon', G_1^*$, such that $\varepsilon'.n = k$, $\varepsilon' = \langle \alpha^{E_1^{**}}, E_2^{**} \rangle$) $(\downarrow W \in \mathcal{S}[[\Xi']] \wedge \varepsilon' \vdash \Xi' \vdash \alpha \sim G_1^*)$, we get that

$$(\downarrow_1 W, \varepsilon'(\langle H_3, \alpha^{H_4} \rangle u_1 :: \alpha) :: G_1^*, \varepsilon'(\langle H_4, \alpha^{E_{22}} \rangle u_2 :: \alpha) :: G_1^*) \in \mathcal{J}_\rho[[G_1^*]]$$

or what is the same $((\langle H_3, \alpha^{H_4} \rangle \circ \varepsilon')$ fails the result follows immediately)

$$(\downarrow_{1+k} W, (\langle H_3, \alpha^{H_4} \rangle \circ \varepsilon') u_1 :: G_1^*, (\langle H_2, \alpha^{H_4} \rangle \circ \varepsilon') u_2 :: G_1^*) \in \mathcal{V}_\rho[[G_1^*]]$$

By definition of transitivity and Lemma A.115, we know that

$$\langle H_3, \alpha^{H_4} \rangle \circ \langle \alpha^{E_1^{**}}, E_2^{**} \rangle = \langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle$$

We know that $\langle E_1^{**}, E_2^{**} \rangle \vdash \Xi' \vdash G'' \sim G_1^*$. Since $\langle E_1^{**}, E_2^{**} \rangle \vdash \Xi \vdash G'' \sim G_1^*$, $\downarrow_1 W \in \mathcal{S}[[\Xi']]$, $(\downarrow_1 W, \langle H_3, H_4 \rangle u_1 :: G'', \langle H_1, H_4 \rangle u_2 :: G'') \in \mathcal{V}_\rho[[G'']]$, by Lemma A.101, we know that (since $(\langle H_3, \alpha^{H_4} \rangle \circ \varepsilon')$ does not fail then $(\langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle)$ also does not fail by the transitivity rules)

$$(\downarrow_{1+k} W, (\langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle) u_1 :: G_1^*, (\langle H_3, H_4 \rangle \circ \langle E_1^{**}, E_2^{**} \rangle) u_2 :: G_1^*) \in \mathcal{V}_\rho[[G_1^*]]$$

The result follows immediately.

3. If $G = ?$ we have the following cases:

- ($G = ?, \varepsilon = \langle H_3, H_4 \rangle$). By the definition of $\mathcal{V}_\rho[[?]]$ in this case we have to prove that:

$$(W, \rho_1(\varepsilon) u_1 :: \rho(G), \rho_2(\varepsilon) u_2 :: \rho(G)) \in \mathcal{V}_\rho[[\text{const}(H_4)]]$$

but as $\text{const}(H_4) = \exists X.?$, we proceed just like the case where $G = \exists X.G'_1$, where $G'_1 = ?$.

- ($G = ?, \varepsilon = \langle H_3, \alpha^{E_4} \rangle$). Notice that as α^{E_4} cannot have free type variables therefore E_3 neither. Then $\varepsilon = \rho_i(\varepsilon)$. By the definition of $\mathcal{V}_\rho[[?]]$ we have to prove that

$$(W, \langle H_3, \alpha^{E_4} \rangle u_1 :: \alpha, \langle H_3, \alpha^{E_4} \rangle u_2 :: \alpha) \in \mathcal{V}_\rho[[\alpha]]$$

Note that by Lemma A.112 we know that $\varepsilon \vdash \Xi; \Delta \vdash G^* \sim \alpha$. Then we proceed just like the case $G \in \text{TYPE NAME}$.

□

Proposition A.150 (Compatibility-Epack) *If $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : G''[G'/X]$, $\Xi; \Delta \vdash G'$, then*

$$\Xi; \Delta; \Gamma \vdash \text{pack}\langle G', t_1 \rangle \text{ as } \exists X.G'' \preceq \text{pack}\langle G', t_2 \rangle \text{ as } \exists X.G'' : \exists X.G''$$

PROOF. First, we are required to prove that

$$\Xi; \Delta; \Gamma \vdash \text{pack}\langle G', t_i \rangle \text{ as } \exists X.G'' : \exists X.G''$$

But by unfolding the premises we know that $\Xi; \Delta; \Gamma \vdash t_i : G''[G'/X]$, therefore:

$$\text{(Epack \& Easc)} \frac{\Xi; \Delta; \Gamma \vdash t_i : G''[G'/X] \quad \Xi; \Delta \vdash G'}{\Xi; \Delta; \Gamma \vdash \text{pack}\langle G', t_i \rangle \text{ as } \exists X.G'' : \exists X.G''}$$

Consider arbitrary W, ρ, γ such that $W \in \mathcal{S}[\Xi]$, $(W, \rho) \in \mathcal{D}[\Delta]$ and $(W, \gamma) \in \mathcal{G}_\rho[\Gamma]$. We are required to show that

$$(W, \rho(\gamma_1(\text{pack}\langle G', t_1 \rangle \text{ as } \exists X.G'')), \rho(\gamma_2(\text{pack}\langle G', t_2 \rangle \text{ as } \exists X.G''))) \in \mathcal{F}_\rho[\exists X.G'']$$

First we have to prove that:

$$W.\Xi_i \vdash \rho(\gamma_i((\text{pack}\langle G', t_i \rangle \text{ as } \exists X.G''))) : \rho(\exists X.G'')$$

As we know that $\Xi; \Delta; \Gamma \vdash (\text{pack}\langle G', t_i \rangle \text{ as } \exists X.G'') : \exists X.G''$, by Lemma A.110 the result follows immediately.

By definition of substitutions

$$\rho(\gamma_i((\text{pack}\langle G', t_i \rangle \text{ as } \exists X.G''))) = (\text{pack}\langle \rho(G'), \rho(\gamma_i(t_i)) \rangle \text{ as } \exists X.\rho(G''))$$

Therefore we have to prove that

$$(W, (\text{pack}\langle \rho(G'), \rho(\gamma_1(t_1)) \rangle \text{ as } \exists X.\rho(G'')), (\text{pack}\langle \rho(G'), \rho(\gamma_2(t_2)) \rangle \text{ as } \exists X.\rho(G''))) \in \mathcal{F}_\rho[\exists X.G'']$$

Second, consider arbitrary $i < W.j, \Xi_1$. Either there exist v_1 such that:

$$W.\Xi_1 \triangleright (\text{pack}\langle \rho(G'), \rho(\gamma_1(t_1)) \rangle \text{ as } \exists X.\rho(G'')) \mapsto^i \Xi_1 \triangleright v_1$$

or

$$W.\Xi_1 \triangleright (\text{pack}\langle \rho(G'), \rho(\gamma_1(t_1)) \rangle \text{ as } \exists X.\rho(G'')) \mapsto^i \mathbf{error}$$

Let us suppose that $W.\Xi_1 \triangleright (\text{pack}\langle \rho(G'), \rho(\gamma_1(t_1)) \rangle \text{ as } \exists X.\rho(G'')) \mapsto^i \Xi_1 \triangleright v_1$. Hence, by inspection of the operational semantics, it follows that there exist $i_1 \leq i$, Ξ_{11} and v_{11} such that:

$$W.\Xi_1 \triangleright (\text{pack}\langle \rho(G'), \rho(\gamma_1(t_1)) \rangle \text{ as } \exists X.\rho(G'')) \mapsto^{i_1} \Xi_{11} \triangleright (\text{pack}\langle \rho(G'), v_{11} \rangle \text{ as } \exists X.\rho(G'')) \mapsto^1$$

$$\Xi_{11} \triangleright \varepsilon_1^p(\text{packu}\langle \rho(G'), v_{11} \rangle \text{ as } \exists X.\rho(G'')) :: \exists X.\rho(G'')$$

where $\varepsilon = \langle \exists X.G'', \exists X.G'' \rangle$ and $\varepsilon_i^p = \rho_i(\varepsilon)$.

We instantiate the hypothesis $\Xi; \Delta; \Gamma \vdash t_1 \leq t_2 : G''[G'/X]$ with W, ρ and γ to obtain that:

$$(W, \rho(\gamma_1(t_1)), \rho(\gamma_2(t_2))) \in \mathcal{J}_\rho[[G''[G'/X]]]$$

We instantiate $\mathcal{J}_\rho[[G''[G'/X]]]$ with i_1, Ξ_{11} and v_{11} (note that $i_1 \leq i < W.j$), hence there exists v_{12} and W_1 , such that $W_1 \succeq W$, $W_1.j = W.j - i_1$, $W.\Xi_2 \triangleright \rho(\gamma_2(t_2)) \mapsto^* W'.\Xi_2 \triangleright v_{12}$, $W'.\Xi_1 = \Xi_{11}$, and $(W_1, v_{11}, v_{12}) \in \mathcal{V}_\rho[[G''[G'/X]]]$ (Note that if $W.\Xi_1 \triangleright \rho(\gamma_1(t_1)) \mapsto^{i_1} \mathbf{error}$ the result follows immediately). Let's take $W' = \downarrow_1 W_1$. Note that we get that $(W', v_{11}, v_{12}) \in \mathcal{V}_\rho[[G''[G'/X]]]$.

Then we have to prove that

$$\begin{aligned} & (W', \varepsilon_1^p(\text{packu}\langle \rho(G'), v_{11} \rangle \text{ as } \exists X.\rho(G'')) :: \exists X.\rho(G''), \\ & \varepsilon_2^p(\text{packu}\langle \rho(G'), v_{12} \rangle \text{ as } \exists X.\rho(G'')) :: \exists X.\rho(G'')) \in \mathcal{V}_\rho[[\exists X.\rho(G'')]] \end{aligned}$$

Let $u_i = \text{packu}\langle \rho(G'), v_{1i} \rangle \text{ as } \exists X.\rho(G'')$ and $\exists X.G_1 = \exists X.G''$, we have to prove that:

$$(W', \varepsilon_1^p u_1 :: \exists X.\rho(G''), \varepsilon_2^p u_2 :: \exists X.\rho(G'')) \in \mathcal{V}_\rho[[\exists X.G'']]$$

1. By the definition of $\mathcal{V}_\rho[[\exists X.G_1]]$, we have to prove that $\forall W'' \succeq W', \alpha.\exists R \in \text{REL}_{W'',j}[\rho(G'), \rho(G'')]$ such that $\forall \varepsilon' \Vdash \Xi; \text{dom}(\rho) \vdash \exists X.G_1' \sim \exists X.G_1' (\varepsilon'.n = l)$ it is true that

$$(W^*, (\rho_1(\varepsilon) \circledast \rho_1(\varepsilon'))[\hat{G}', \hat{\alpha}]v_{11} :: \rho(G_1')[\alpha/X], (\rho_2(\varepsilon) \circledast \rho_2(\varepsilon'))[\hat{G}_2, \hat{\alpha}]v_{12} :: \rho(G_1')[\alpha/X]) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[[G_1']]$$

where $W^* = W'' \boxtimes (\alpha, \rho(G'), \rho(G''), R)$.

or what is the same, we have to prove that

$$(W^*, (\rho_1(\varepsilon \circledast \varepsilon'))[\hat{G}', \hat{\alpha}]v_{11} :: \rho(G_1')[\alpha/X], (\rho_2(\varepsilon \circledast \varepsilon'))[\hat{G}_2, \hat{\alpha}]v_{12} :: \rho(G_1')[\alpha/X]) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[[G_1']]$$

By Proposition A.155 (decomposition of the evidence) we know that

$$\rho_i(\varepsilon \circledast \varepsilon')[\hat{G}', \hat{\alpha}] = \pi_1^*(\rho_i(\varepsilon \circledast \varepsilon'))[\hat{G}', \hat{\alpha}] \circledast \rho_i(\varepsilon \circledast \varepsilon')[\hat{\alpha}, \hat{\alpha}]$$

Lets take $R = \mathcal{V}_\rho[[G']]$.

Note that

- $W^* = W'' \boxtimes (\alpha, \rho(G'), \rho(G''), \mathcal{V}_\rho[[G']]) \succeq W'$
- $E'_i = \text{lift}_{W^*.\Xi_i}(\rho(G'))$,
- $E_{i*} = \text{lift}_{W^*.\Xi_i}(G_{pi}), G_{pi} = \text{unlift}(\pi_1(\rho_i(\varepsilon \circledast \varepsilon'))) \sqsubseteq \rho(G'')$,
- $\rho' = \rho[X \mapsto \alpha]$,
- $\varepsilon_i^{-1} = \pi_1^*(\rho_i(\varepsilon \circledast \varepsilon'))[\hat{G}', \hat{\alpha}] = \langle E_{i*}[E'_i/X], E_{i*}[\alpha^{E_i}/X] \rangle$, such that $\varepsilon_i^{-1} \Vdash W^*.\Xi_i \vdash \rho(G''[G'/X]) \sim \rho(G''[\alpha/X])$, $\alpha^{E_i} = \text{lift}_{W^*.\Xi_i}(\alpha)$, and $E'_i = \text{lift}_{W^*.\Xi_i}(\rho(G'))$, $\varepsilon_i^{-1}.n = k$ and
- $(W', v_{11}, v_{12}) \in \mathcal{V}_\rho[[G''[G'/X]]]$, then $(W^*, v_{11}, v_{12}) \in \mathcal{V}_\rho[[G''[G'/X]]]$.

By the Lemma A.153 (compositionality) we know that

$$(W^*, \pi_1(\rho_1(\varepsilon \circledast \varepsilon'))[\hat{G}', \hat{\alpha}]v_{11} :: \rho'(G''), \pi_1(\rho_2(\varepsilon \circledast \varepsilon'))[\hat{G}', \hat{\alpha}]v_{12} :: \rho'(G'')) \in \mathcal{J}_{\rho'}[[G'']]$$

or what is the same

$$(W^*, \pi_1^*(\rho_1(\varepsilon \circledast \varepsilon'))[\hat{G}', \hat{\alpha}]v_{11} :: \rho(G'')[\alpha/X], \pi_1^*(\rho_2(\varepsilon \circledast \varepsilon'))[\hat{G}', \hat{\alpha}]v_{12} :: \rho(G'')[\alpha/X]) \in \mathcal{J}_{\rho[\alpha/X]}[[G'']]$$

Then we know that

$$(\downarrow_k W^*, \varepsilon'_1 u'_1 :: \rho(G'')[\alpha/X], \varepsilon'_2 u'_2 :: \rho(G'')[\alpha/X]) \in \mathcal{V}_{\rho[\alpha/X]}[[G'']]$$

where $v_{1i} = \varepsilon'_{1i} u'_i :: \rho(G''[G'/X])$ and $\varepsilon'_i = \varepsilon'_{1i} \circledast \pi_1^*(\rho_i(\varepsilon \circledast \varepsilon'))[\hat{G}', \hat{\alpha}]$.

Note now that

- $(\downarrow_k W^*, \varepsilon'_1 u'_1 :: \rho(G'')[\alpha/X], \varepsilon'_2 u'_2 :: \rho(G'')[\alpha/X]) \in \mathcal{V}_{\rho[\alpha/X]}[[G'']]$,
- $(\varepsilon \circledast \varepsilon')[X] \Vdash \Xi; \Delta, X \vdash G'' \sim G'_1, (\varepsilon \circledast \varepsilon')[X].n = l$
- $\downarrow_k W^* \in \mathcal{S}[[\Xi]]$ and $(\downarrow_k W^*, \rho') \in \mathcal{D}[[\Delta, X]]$,

Then, by Lemma A.152 (Ascription Lemma), we know that

$$(\downarrow_{k+l} W^*, (\varepsilon'_1 \circledast \rho'_1((\varepsilon \circledast \varepsilon')[X]))u'_1 :: \rho'(G'_1), (\varepsilon'_2 \circledast \rho'_2((\varepsilon \circledast \varepsilon')[X]))u'_2 :: \rho'(G'_1)) \in \mathcal{V}_{\rho'}[[G'_1]]$$

or what is the same

$$(\downarrow_{k+l} W^*, (\varepsilon'_1 \circledast \rho_1(\varepsilon \circledast \varepsilon')[\hat{\alpha}, \hat{\alpha}])u'_1 :: \rho(G'_1)[\alpha/X], (\varepsilon'_2 \circledast \rho_2(\varepsilon \circledast \varepsilon')[\hat{\alpha}, \hat{\alpha}])u'_2 :: \rho(G'_1)[\alpha/X]) \in \mathcal{V}_{\rho[\alpha/X]}[[G'_1]]$$

The result follows immediately.

$$(W^*, (\rho_1(\varepsilon \circledast \varepsilon'))[\hat{G}', \hat{\alpha}]v_{11} :: \rho(G'_1)[\alpha/X], (\rho_2(\varepsilon \circledast \varepsilon'))[\hat{G}_2, \hat{\alpha}]v_{12} :: \rho(G'_1)[\alpha/X]) \in \mathcal{J}_{\rho[\alpha/X]}[[G'_1]]$$

□

Proposition A.151 (Compatibility-Eunpack) *If $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : \exists X.G_1$, $\Xi; \Delta, X; \Gamma, x : G_1 \vdash t'_1 \preceq t'_2 : G_2$ and $\Xi; \Delta \vdash G_2$, then $\Xi; \Delta; \Gamma \vdash \text{unpack}\langle X, x \rangle = t_1$ in $t'_1 \preceq \text{unpack}\langle X, x \rangle = t_2$ in $t'_2 : G_2$.*

PROOF. First, we are required to prove that

$$\Xi; \Delta; \Gamma \vdash \text{unpack}\langle X, x \rangle = t_i \text{ in } t'_i : G_2$$

But by unfolding the premises we know that $\Xi; \Delta; \Gamma \vdash t_i : \exists X.G_1$, $\Xi; \Delta, X; \Gamma, x : G_1 \vdash t'_i : G_2$ and $\Xi; \Delta \vdash G_2$, therefore:

$$\text{(Eunpack)} \frac{\Xi; \Delta; \Gamma \vdash t_i : \exists X.G_1 \quad \Xi; \Delta, X; \Gamma, x : G_1 \vdash t'_i : G_2 \quad \Xi; \Delta \vdash G_2}{\Xi; \Delta; \Gamma \vdash \text{unpack}\langle X, x \rangle = t_i \text{ in } t'_i : G_2}$$

Consider arbitrary W, ρ, γ such that $W \in \mathcal{S}[\Xi]$, $(W, \rho) \in \mathcal{D}[\Delta]$ and $(W, \gamma) \in \mathcal{G}_\rho[\Gamma]$. We are required to show that

$$(W, \rho(\gamma_1(\text{unpack}\langle X, x \rangle = t_1 \text{ in } t'_1)), \rho(\gamma_2(\text{unpack}\langle X, x \rangle = t_2 \text{ in } t'_2))) \in \mathcal{T}_\rho[G_2]$$

First we have to prove that:

$$W.\Xi_i \vdash \rho(\gamma_i(\text{unpack}\langle X, x \rangle = t_i \text{ in } t'_i)) : \rho(G_2)$$

As we know that $\Xi; \Delta; \Gamma \vdash \text{unpack}\langle X, x \rangle = t_i \text{ in } t'_i : G_2$, by Lemma A.110 the result follows immediately.

By definition of substitutions

$$\rho(\gamma_i(\text{unpack}\langle X, x \rangle = t_i \text{ in } t'_i)) = \text{unpack}\langle X, x \rangle = \rho(\gamma_i(t_i)) \text{ in } \rho(\gamma_i(t'_i))$$

Therefore we have to prove that

$$(W, \text{unpack}\langle X, x \rangle = \rho(\gamma_1(t_1)) \text{ in } \rho(\gamma_1(t'_1)), \text{unpack}\langle X, x \rangle = \rho(\gamma_2(t_2)) \text{ in } \rho(\gamma_2(t'_2))) \in \mathcal{T}_\rho[G_2]$$

Second, consider arbitrary $i < W.j, \Xi_1$. Either there exist v_1 such that:

$$W.\Xi_1 \triangleright \text{unpack}\langle X, x \rangle = \rho(\gamma_1(t_1)) \text{ in } \rho(\gamma_1(t'_1)) \mapsto^i \Xi_1 \triangleright v_1$$

or

$$W.\Xi_1 \triangleright \text{unpack}\langle X, x \rangle = \rho(\gamma_1(t_1)) \text{ in } \rho(\gamma_1(t'_1)) \mapsto^i \Xi_1 \triangleright \mathbf{error}$$

Let us suppose that $W.\Xi_1 \triangleright \text{unpack}\langle X, x \rangle = \rho(\gamma_1(t_1)) \text{ in } \rho(\gamma_1(t'_1)) \mapsto^i \Xi_1 \triangleright v_1$.

Hence, by inspection of the operational semantics, it follows that there exist $i_1 \leq i, \Xi_{11}$ and v_{11} such that:

$$W.\Xi_1 \triangleright \rho(\gamma_1(t_1)) \mapsto^{i_1} \Xi_{11} \triangleright v_{11}$$

Instantiate the second conjunct of $\Xi; \Delta; \Gamma \vdash t_1 \preceq t_2 : \exists X.G_1$ with W, ρ , and γ . Note that $W \in \mathcal{S}[\Xi]$, $(W, \rho) \in \mathcal{D}[\Delta]$ and $(W, \gamma) \in \mathcal{G}_\rho[\Gamma]$. Then we have that $(W, \rho(\gamma_1(t_1)), \rho(\gamma_2(t_2))) \in \mathcal{T}_\rho[\exists X.G_1]$. Instantiate this with i_1, Ξ_{11} and v_{11} . Note that $i_1 < W.j$ which follows from $i_1 \leq i < W.j$.

Hence, there exists $W_1 \succeq W$ and v_{12} such that $W.\Xi_2 \triangleright \rho(\gamma_2(t_2)) \mapsto^* W_1.\Xi_2 \triangleright v_{12}$, $(W_1, v_{11}, v_{12}) \in \mathcal{V}_\rho[\exists X.G_1]$ and $W_1.j + i_1 = W.j$.

Hence, $v_{1i} = \varepsilon'_i(\text{packu}\langle G'_i, v'_i \rangle \text{ as } \exists X.G''_i) :: \exists X.\rho(G_1)$, where $\varepsilon'_1 = k.n$ and $v'_i = \varepsilon_{pi}u_i :: G_{pi}$.

From $(W_1, v_{11}, v_{12}) \in \mathcal{V}_\rho[\exists X.G_1]$, it follows that there exists $R \in \text{REL}_{W_1.j}[G'_1, G'_2]$ such that $\forall \varepsilon' \Vdash \Xi; \Delta \vdash \exists X.G_1 \sim \exists X.G_1 (\varepsilon'.n = l)$ it is true that

$$(W'_1, (\varepsilon'_1 \circ \rho_1(\varepsilon'))[\hat{G}'_1, \hat{\alpha}]v'_1 :: \rho(G_1)[\alpha/X], (\varepsilon'_2 \circ \rho_2(\varepsilon'))[\hat{G}'_2, \hat{\alpha}]v'_2 :: \rho(G_1)[\alpha/X]) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[G_1]$$

where $W'_1 = W_1 \boxtimes (\alpha, G'_1, G'_2, R)$. If we take $\varepsilon' = \mathcal{J}(\Xi, ()\exists X.G_1, \exists X.G_1)$, then

$$(\varepsilon'_i \circ \rho_i(\varepsilon')) = \varepsilon'_i$$

Therefore we know that

$$(W'_1, \varepsilon'_1[\hat{G}'_1, \hat{\alpha}]v'_1 :: \rho(G_1)[\alpha/X], \varepsilon'_2[\hat{G}'_2, \hat{\alpha}]v'_2 :: \rho(G_1)[\alpha/X]) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[G_1]$$

If $W'_1.\Xi_1 \triangleright \varepsilon'_1[\hat{G}'_1, \hat{\alpha}]v'_1 :: \rho(G_1)[\alpha/X] \mapsto \mathbf{error}$ the result follows immediately. Otherwise, if

$$W'_1.\Xi_1 \triangleright \varepsilon'_1[\hat{G}'_1, \hat{\alpha}]v'_1 :: \rho(G_1)[\alpha/X] \mapsto^{k+l} W'_1.\Xi_1 \triangleright v_{p1}$$

where $v_{p1} = (\varepsilon_{p1} \circ \varepsilon'_1[\hat{G}'_1, \hat{\alpha}]u_1 :: \rho(G_1)[\alpha/X])$, then

$$W'_1.\Xi_2 \triangleright \varepsilon'_2[\hat{G}'_2, \hat{\alpha}]v'_2 :: \rho(G_1)[\alpha/X] \mapsto^* W'_1.\Xi_2 \triangleright v_{p2}$$

where $v_{p2} = (\varepsilon_{p2} \circ \varepsilon'_2[\hat{G}'_2, \hat{\alpha}]u_2 :: \rho(G_1)[\alpha/X])$ and $(W'_2, v_{p1}, v_{p2}) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[G_1]$, where $W'_2 = \downarrow_{k+l} W'_1$ and $W'_2.j + k + l = W'_1.j$.

Note that

$$W.\Xi_1 \triangleright \mathbf{unpack}\langle X, x \rangle = \rho(\gamma_1(t_1)) \text{ in } \rho(\gamma_1(t'_1)) \mapsto^{i_1}$$

$$W_1.\Xi_1 \triangleright \mathbf{unpack}\langle X, x \rangle = v_{11} \text{ in } \rho(\gamma_1(t'_1)) \mapsto^{k+l} W_1.\Xi_1 \triangleright t_2[\alpha/X][v_{p1}/x] \mapsto^{i_2} \Xi_1 \triangleright v_1$$

where $i = i_1 + k + l + i_2$.

Instantiate the second conjunct of $\Xi; \Delta, X; \Gamma, x : G_1 \vdash t'_1 \preceq t'_2 : G_2$ with $W'_2, \rho[X \mapsto \alpha], \gamma[x \mapsto (v_{p1}, v_{p2})]$. Note that $W'_2 \in \mathcal{S}[\Xi](W'_2 \succeq W)$, $(W'_2, \rho[X \mapsto \alpha]) \in \mathcal{D}[\Delta, X]$ and $(W'_2, \gamma[x \mapsto (v_{p1}, v_{p2})]) \in \mathcal{G}_\rho[\Gamma, x : G_1]$. Then we have that

$$(W'_2, \gamma_1(\rho(t'_1))[\hat{\alpha}/X][v_{p1}/x], \gamma_2(\rho(t'_2))[\hat{\alpha}/X][v_{p2}/x]) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[G_2]$$

Instantiate this with $i_2 < W'_2.j = W.j - i_1 - k - l$ ($i_2 = i - i_1 - k - l, i < W.j$), Ξ_1 and v_1 . Hence, there exists $W_2 \succeq W'_2$ and v_2 such that

$$W'.\Xi_2 \triangleright \gamma_2(\rho(t'_2))[\hat{\alpha}/X][v_{p2}/x] \mapsto^* W_2.\Xi_2 \triangleright v_2, W_2.\Xi_1 = \Xi_1, W_2.j + i_2 = W'_2.j \text{ and}$$

$$(W_2, v_1, v_2) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[G_2]$$

We are required to show that there exists $W_2 \succeq W$ and v_2 , such that

$$W.\Xi_2 \triangleright \mathbf{unpack}\langle X, x \rangle = \rho(\gamma_2(t_2)) \text{ in } \rho(\gamma_2(t'_2)) \mapsto^* W_2.\Xi_2 \triangleright v_2$$

, $W_2.j + i = W.j$ ($W_2.j = W.j - i_1 - k - l - i_2, i = i_1 + k + l + i_2$) and $(W_2, v_1, v_2) \in \mathcal{V}_\rho[G_2]$, which follows from $(W_2, v_1, v_2) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[G_2]$ and $\Xi; \Delta \vdash G_2$. \square

Proposition A.152 (Ascriptions Preserve Relations) *If $(W, v_1, v_2) \in \mathcal{V}_\rho[[G]]$, $\varepsilon \Vdash \Xi$; $\Delta \vdash G \sim G'$, $W \in \mathcal{S}[[\Xi]]$ and $(W, \rho) \in \mathcal{D}[[\Delta]]$, then $(W, \rho_1(\varepsilon)v_1 :: \rho(G'), \rho_2(\varepsilon)v_2 :: \rho(G')) \in \mathcal{T}_\rho[[G']]$.*

PROOF. We only prove the case for existential, the other cases are in A.6.2.

Case $(G = \exists X.G''_1$ and $G' = \exists X.G'_1$). We know that

$$(W, v_1, v_2) \in \mathcal{V}_\rho[[\exists X.G''_1]]$$

Where $v_i = \varepsilon_i(\text{packu}\langle G_i^*, v'_i \rangle \text{ as } \exists X.\rho(G_i''')) :: \exists X.\rho(G''_1)$ and $\varepsilon_i \vdash W.\Xi_i \vdash \exists X.\rho(G_i''') \sim \exists X.\rho(G''_1)$. Let's suppose that $\rho_1(\varepsilon).n = k$ and $\varepsilon_1.n = m$. We have to prove that

$$(W, \rho_1(\varepsilon)v_1 :: \exists X.\rho(G'_1), \rho_2(\varepsilon)v_2 :: \exists X.\rho(G'_1)) \in \mathcal{T}_\rho[[\exists X.G'_1]]$$

If $(\varepsilon_1 \circ \rho_1(\varepsilon))$ fails, then we apply Lemma A.111 to show that $(\varepsilon_2 \circ \rho_2(\varepsilon))$ also fails, therefore the proof holds immediately. In the other case, $(\varepsilon_i \circ \rho_i(\varepsilon))$ do not fail, then by the definition of $\mathcal{T}_\rho[[\exists X.G'_1]]$, we have to prove that:

$$\begin{aligned} (\downarrow_k W, (\varepsilon_1 \circ \rho_1(\varepsilon))(\text{packu}\langle G_1^*, v'_1 \rangle \text{ as } \exists X.\rho(G_1''')) :: \exists X.\rho(G'_1), (\varepsilon_2 \circ \rho_2(\varepsilon))(\text{packu}\langle G_2^*, v'_2 \rangle \text{ as } \exists X.\rho(G_2''')) :: \exists X.\rho(G'_1)) \\ \in \mathcal{V}_\rho[[\exists X.G'_1]] \end{aligned}$$

or what is the same:

$$\begin{aligned} \forall W'' \succeq \downarrow_k W, \alpha. \exists R \in \text{REL}_{W''}.j[G_1^*, G_2^*]. \\ (W''.\Xi_1 \vdash G_1^* \wedge W''.\Xi_2 \vdash G_2^* \wedge \forall \Xi, \varepsilon' \Vdash \Xi; \text{dom}(\rho) \vdash \exists X.G'_1 \sim \exists X.G'_1, \Xi \in \mathcal{S}[[\Xi]], \varepsilon'.n = l. \\ (W''', (\varepsilon_1 \circ \rho_1(\varepsilon \circ \varepsilon'))[\hat{G}_1^*, \hat{\alpha}]v'_1 :: \rho(G'_1)[\alpha/X], (\varepsilon_2 \circ \rho_2(\varepsilon \circ \varepsilon'))[\hat{G}_2^*, \hat{\alpha}]v'_2 :: \rho(G'_1)[\alpha/X]) \in \mathcal{T}_{\rho[X \mapsto \alpha]}[[G'_1]] \end{aligned}$$

where $W''' = ((W'') \boxtimes (\alpha, G_1^*, G_2^*, R))$.

Let's suppose that $v'_i = \varepsilon_i^* u_i :: G_i'''[G_i^*]$. Therefore, we are required to prove that

$$\begin{aligned} ((\downarrow_{k+l+m} W''')(\varepsilon_1^* \circ \varepsilon_1 \circ \rho_1(\varepsilon \circ \varepsilon'))[\hat{G}_1^*, \hat{\alpha}]u_1 :: \rho(G'_1)[\alpha/X], \\ \varepsilon_2^* \circ (\varepsilon_2 \circ \rho_2(\varepsilon \circ \varepsilon'))[\hat{G}_2^*, \hat{\alpha}]u_2 :: \rho(G'_1)[\alpha/X]) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[[G'_1]] \end{aligned}$$

Note that by Lemma A.156 we get that

$$\begin{aligned} (\varepsilon_i \circ \rho_i(\varepsilon \circ \varepsilon'))[\hat{G}_i^*, \hat{\alpha}] = (\varepsilon_i \circ \pi_1^2(\rho_i(\varepsilon \circ \varepsilon')))[\hat{G}_i^*, \hat{\alpha}] \circ \rho_i(\varepsilon \circ \varepsilon')[\hat{\alpha}, \hat{\alpha}] = \\ (\varepsilon_i \circ \rho_i(\pi_1^2(\varepsilon \circ \varepsilon')))[\hat{G}_i^*, \hat{\alpha}] \circ \rho_i(\varepsilon \circ \varepsilon')[\hat{\alpha}, \hat{\alpha}] \end{aligned}$$

By premise, we know that $(W, v_1, v_2) \in \mathcal{V}_\rho[[\exists X.G''_1]]$. Then, we instantiate this definition with $(\uparrow_k W'') \succeq W$ ($W'' \succeq (\downarrow_k W) \Rightarrow (\uparrow_k W'') \succeq \uparrow_k \downarrow_k W$) and α . Therefore, $\exists R \in \text{REL}_{W''}.j[G_1^*, G_2^*]$, such that for all evidence $\varepsilon'' \Vdash \Xi'$; $\text{dom}(\rho) \vdash \exists X.G''_1 \sim \exists X.G''_1$, in particular $\varepsilon'' = \pi_1^2(\varepsilon \circ \varepsilon')$ ($\pi_1^2(\varepsilon \circ \varepsilon').n = k$). Therefore, we know that $(W''' = (W'' \boxtimes (\alpha, G_1^*, G_2^*, R)))$:

$$(W''', (\varepsilon_1 \circ \rho_1(\pi_1^2(\varepsilon \circ \varepsilon'))))[\hat{G}_1^*, \hat{\alpha}]v_1' :: \rho(G_1'')[\alpha/X], (\varepsilon_2 \circ \rho_2(\pi_1^2(\varepsilon \circ \varepsilon')))[\hat{G}_2^*, \hat{\alpha}]v_2' :: \rho(G_1'')[\alpha/X] \\ \in \mathcal{J}_{\rho[X \mapsto \alpha]}[\![G_1'']\!]$$

Then, we get that:

$$((\downarrow_{k+l}W'''), v_1''', v_2''') \in \mathcal{V}_{\rho[X \mapsto \alpha]}[\![G_1'']\!]$$

where $v_i''' = \varepsilon_i^* \circ (\varepsilon_i \circ \rho_i(\pi_1^2(\varepsilon \circ \varepsilon')))[\hat{G}_i^*, \hat{\alpha}]u_i :: \rho(G_1'')[\alpha/X]$.

By induction hypothesis on $((\downarrow_{k+l}W'''), v_1''', v_2''') \in \mathcal{V}_{\rho[X \mapsto \alpha]}[\![G_1'']\!]$, with $(\varepsilon \circ \varepsilon')[X] \Vdash \Xi; \Delta, X \vdash G_1'' \sim G_1' ((\varepsilon \circ \varepsilon')[X].n = m)$, $(\downarrow_{k+l}W''') \in \mathcal{S}[\![\Xi]\!]$ and $((\downarrow_{k+l}W'''), \rho') \in \mathcal{D}[\![\Delta, X]\!]$, $\rho' = \rho[X \mapsto \alpha]$, we get that:

$$((\downarrow_{k+l}W'''), \rho'_1((\varepsilon \circ \varepsilon')[X])v_1''' :: \rho(G_1')[\alpha/X], \rho'_2((\varepsilon \circ \varepsilon')[X])v_2''' :: \rho(G_1')[\alpha/X]) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[\![G_1']\!]$$

or what is the same (note that $\rho'_i((\varepsilon \circ \varepsilon')[X]) = \rho_i(\varepsilon \circ \varepsilon')[\hat{\alpha}, \hat{\alpha}]$):

$$((\downarrow_{k+l}W'''), \rho_1(\varepsilon \circ \varepsilon')[\hat{\alpha}, \hat{\alpha}]v_1''' :: \rho(G_1')[\alpha/X], \rho_2(\varepsilon \circ \varepsilon')[\hat{\alpha}, \hat{\alpha}]v_2''' :: \rho(G_1')[\alpha/X]) \in \mathcal{J}_{\rho[X \mapsto \alpha]}[\![G_1']\!]$$

or what is the same:

$$((\downarrow_{k+l+m}W'''), v_1^*, v_2^*) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[\![G_1']\!]$$

where $v_i^* = \varepsilon_i^* \circ ((\varepsilon_i \circ \rho_i(\pi_1^2(\varepsilon \circ \varepsilon')))[\hat{G}_i^*, \hat{\alpha}] \circ \rho_i(\varepsilon \circ \varepsilon')[\hat{\alpha}, \hat{\alpha}])u_i :: \rho(G_1')[\alpha/X]$.

By the reduction rule

$$W'''.\Xi_1 \triangleright (\varepsilon_1 \circ \rho_1(\varepsilon \circ \varepsilon'))[\hat{G}_1^*, \hat{\alpha}]v_1' :: \rho(G_1')[\alpha/X] \longrightarrow^{k+m+l} W'''.\Xi_1 \triangleright v_1^*$$

Therefore, the results follows immediately $((\downarrow_{k+l+m}W'''), v_1^*, v_2^*) \in \mathcal{V}_{\rho[X \mapsto \alpha]}[\![G_1']\!]$.

□

Proposition A.153 (CompositionalityEx) *If*

- $W.\Xi_i(\alpha) = \rho(G')$ and $W.\kappa(\alpha) = \mathcal{V}_\rho[\![G']\!]$,
- $E_i' = \text{lift}_{W.\Xi_i}(\rho(G'))$,
- $E_i = \text{lift}_{W.\Xi_i}(G_p)$ for some $G_p \sqsubseteq \rho(G)$,
- $\rho' = \rho[X \mapsto \alpha]$,
- $\varepsilon_i = \langle E_i[\alpha^{E_i}/X], E_i[E_i'/X] \rangle$, such that $\varepsilon_i \vdash W.\Xi_i \vdash \rho(G[\alpha/X]) \sim \rho(G[G'/X])$, and
- $\varepsilon_i^{-1} = \langle E_i[E_i'/X], E_i[\alpha^{E_i}/X] \rangle$, such that $\varepsilon_i^{-1} \vdash W.\Xi_i \vdash \rho(G[G'/X]) \sim \rho(G[\alpha/X])$, then

1.

$$(W, \varepsilon'_1 u_1 :: \rho'(G), \varepsilon'_2 u_2 :: \rho'(G)) \in \mathcal{V}_{\rho'} \llbracket G \rrbracket \Rightarrow \\ (W, \varepsilon_1(\varepsilon'_1 u_1 :: \rho(G)) :: \rho(G[G'/X]), \varepsilon_2(\varepsilon'_2 u_2 :: \rho(G)) :: \rho(G[G'/X])) \in \mathcal{T}_{\rho} \llbracket G[G'/X] \rrbracket$$

2.

$$(W, \varepsilon'_1 u_1 :: \rho(G[G'/X]), \varepsilon'_2 u_2 :: \rho(G[G'/X])) \in \mathcal{V}_{\rho} \llbracket G[G'/X] \rrbracket \Rightarrow \\ (W, \varepsilon_1^{-1}(\varepsilon'_1 u_1 :: \rho(G[G'/X])) :: \rho'(G), \varepsilon_2^{-1}(\varepsilon'_2 u_2 :: \rho(G[G'/X])) :: \rho'(G)) \in \mathcal{T}_{\rho'} \llbracket G \rrbracket$$

PROOF. We only prove the case for existential, the other cases are in A.6.2. We proceed by induction on G . Let $v_i = \varepsilon'_i u_i :: \rho'(G)$, $\Delta = \text{dom}(\rho)$. We prove (1) first. Let's suppose that $\varepsilon'_1.n = k$, $\varepsilon_1.n = l$ and $\varepsilon_1^{-1}.n = m$.

Case $(\exists Y.G_1)$. We know that

$$(W, \varepsilon'_1 u_1 :: \rho'(G), \varepsilon'_2 u_2 :: \rho'(G)) \in \mathcal{V}_{\rho'} \llbracket G \rrbracket$$

where $u_i = \text{packu}\langle G_i^*, v'_i \rangle$ as $\exists Y.G_i''$ and $G = \exists Y.G_1$. Therefore, we have to prove that

$$(W, \varepsilon_1(\varepsilon'_1 u_1 :: \rho(G)) :: \rho(G[G'/X]), \varepsilon_2(\varepsilon'_2 u_2 :: \rho(G)) :: \rho(G[G'/X])) \in \mathcal{V}_{\rho} \llbracket G[G'/X] \rrbracket$$

If $\varepsilon'_i \circ \varepsilon_i$ is not defined, the result follows immediately. If it is defined, we have to prove that:

$$((\downarrow_l W), (\varepsilon'_1 \circ \varepsilon_1) u_1 :: \rho(G[G'/X]), (\varepsilon'_2 \circ \varepsilon_2) u_2 :: \rho(G[G'/X])) \in \mathcal{V}_{\rho} \llbracket G[G'/X] \rrbracket$$

or what is the same by the definition of $\mathcal{V}_{\rho} \llbracket G[G'/X] \rrbracket$, we have to prove that:

$$\forall W'' \succeq (\downarrow_l W), \beta. \exists R \in \text{REL}_{W'', j}[G_1^*, G_2^*]. \\ (W''. \Xi_1 \vdash G_1^* \wedge W''. \Xi_2 \vdash G_2^* \wedge \forall \varepsilon' \Vdash \Xi; \Delta \vdash \exists Y.G_1[G'/X] \sim \exists Y.G_1[G'/X] \wedge \varepsilon'.n = k' \\ (W''', (\varepsilon'_1 \circ \varepsilon_1 \circ \rho_1(\varepsilon'))[\hat{G}_1^*, \hat{\beta}] v'_1 :: \rho(G_1[G'/X][\beta/Y]), (\varepsilon'_2 \circ \varepsilon_2 \circ \rho_2(\varepsilon'))[\hat{G}_2^*, \hat{\beta}] v'_2 :: \rho(G_1[G'/X][\beta/Y])) \\ \in \mathcal{T}_{\rho[Y \mapsto \beta]} \llbracket G_1[G'/X] \rrbracket$$

where $W''' = ((W'') \boxtimes (\alpha, G_1^*, G_2^*, R))$. Therefore, we are required to prove that

$$((\downarrow_{k+l+k'} W''') (\varepsilon_1^* \circ (\varepsilon'_1 \circ \varepsilon_1 \circ \rho_1(\varepsilon'))[\hat{G}_1^*, \hat{\beta}]) u'_1 :: \rho(G_1[G'/X][\beta/Y]), \\ (\varepsilon_2^* \circ (\varepsilon'_2 \circ \varepsilon_2 \circ \rho_2(\varepsilon'))[\hat{G}_2^*, \hat{\beta}]) u'_2 :: \rho(G_1[G'/X][\beta/Y])) \in \mathcal{T}_{\rho[Y \mapsto \beta]} \llbracket G_1[G'/X] \rrbracket$$

where $v'_i = \varepsilon_i^* u'_i :: G_i''[G_i^*/Y]$.

Note that by Lemma A.157 we know that $\varepsilon_i = \rho_i(\varepsilon^{**})[\alpha, \rho(G'), X]$ for some $\varepsilon^{**} \Vdash \Xi; \Delta, X \vdash \exists Y.G_1 \sim \exists Y.G_1$. Therefore, by Lemma A.158 we get that for some $\varepsilon^* \Vdash \Xi; \Delta, X \vdash \exists Y.G_1 \sim \exists Y.G_1$:

$$(\varepsilon'_i \circ \varepsilon_i \circ \rho_i(\varepsilon'))[\hat{G}_i^*, \hat{\beta}] = (\varepsilon'_i \circ \rho_i(\varepsilon^{**})[\alpha, \rho(G'), X] \circ \rho_i(\varepsilon'))[\hat{G}_i^*, \hat{\beta}] =$$

$$\begin{aligned}
& (\varepsilon'_i \circ \rho_i(\varepsilon^*)[\alpha, \alpha, X])[\hat{G}_i^*, \hat{\beta}] \circ (\pi_2^*(\rho_i(\varepsilon^*))[\alpha, \rho(G'), Y] \circ \rho_i(\varepsilon'))[\hat{\beta}, \hat{\beta}] = \\
& (\varepsilon'_i \circ \rho'_i(\varepsilon^*))[\hat{G}_i^*, \hat{\beta}] \circ (\pi_2^*(\rho_i(\varepsilon^*))[\alpha, \rho(G'), Y] \circ \rho_i(\varepsilon'))[\hat{\beta}, \hat{\beta}]
\end{aligned}$$

By premise, we know that $(W, \varepsilon'_1 u_1 :: \rho'(G), \varepsilon'_2 u_2 :: \rho'(G)) \in \mathcal{V}_{\rho'}[\llbracket \exists Y.G_1 \rrbracket]$. Then, we instantiate this definition with $(\uparrow_l W''') \succeq W$ ($W''' \succeq (\downarrow_l W) \Rightarrow \uparrow W'' \succeq (\uparrow_l \downarrow_l W)$) and β . Therefore, $\exists R \in \text{REL}_{W'', j}[G_1^*, G_2^*]$, such that for all evidence $\varepsilon'' \Vdash \Xi; \Delta, X \vdash \exists X.G'_1 \sim \exists X.G'_1$, in particular, we instantiate with $\varepsilon'' = \varepsilon^*[X]$ ($\varepsilon''.n = l$). Therefore, we know that $(W''' = ((W''') \boxtimes (\beta, G_1^*, G_2^*, R)))$:

$$(\uparrow_l W''', (\varepsilon'_1 \circ \rho'_1(\varepsilon^*))[\hat{G}_1^*, \hat{\beta}] v'_1 :: \rho'(G_1)[\beta/Y], (\varepsilon'_2 \circ \rho'_2(\varepsilon^*))[\hat{G}_2^*, \hat{\beta}] v'_2 :: \rho'(G_1)[\beta/Y]) \in \mathcal{J}_{\rho'[Y \mapsto \beta]}[\llbracket G_1 \rrbracket]$$

Therefore, we know that

$$(\downarrow_k W''', v''_1, v''_2) \in \mathcal{V}_{\rho'[Y \mapsto \beta]}[\llbracket G_1 \rrbracket]$$

where $v''_i = \varepsilon_i^* \circ (\varepsilon'_i \circ \rho'_i(\varepsilon^*))[\hat{G}_i^*, \hat{\beta}] u'_i :: \rho'(G_1)[\beta/Y]$.

Note that, for some $G_{ph} \sqsubseteq \rho[Y \mapsto \beta](G_1)$, we get $E_i^* = \text{lift}_{W''', \Xi_i}(G_{ph})$ such that:

- $\text{unlift}(\pi_2(\rho[Y \mapsto \beta]_i(\varepsilon^*))) = G_{ph} \sqsubseteq \rho[Y \mapsto \beta](G_1)$ and $E_i^* = \text{lift}_{W''', \Xi_i}(G_{ph})$
- $\pi_2^*(\rho_i(\varepsilon^*))[\hat{\beta}, \hat{\beta}] = \pi_2^*(\rho[Y \mapsto \beta]_i(\varepsilon^*)) = \langle E_i^*, E_i^* \rangle$, by the definition of $\pi_2^*(\cdot)[\cdot]$
- $\pi_2^*(\rho_i(\varepsilon^*))[\alpha, \rho(G'), X][\hat{\beta}, \hat{\beta}] = \pi_2^*(\rho[Y \mapsto \beta]_i(\varepsilon^*))[\alpha, \rho(G'), X]$
- $\langle E_i^*[\alpha^{E'_i}/X], E_i^*[E'_i/X] \rangle = \langle E_i^*, E_i^* \rangle[\alpha, \rho(G'), X] = \pi_2^*(\rho_i(\varepsilon^*))[\alpha, \rho(G'), X][\hat{\beta}, \hat{\beta}]$

Now, by the induction hypothesis we get:

- $(\downarrow_k W''', v''_1, v''_2) \in \mathcal{V}_{\rho'[Y \mapsto \beta]}[\llbracket G_1 \rrbracket]$
- $W_i''', \Xi(\alpha) = \rho[Y \mapsto \beta](G')$ and $W.\kappa(\alpha) = \mathcal{V}_{\rho[Y \mapsto \beta]}[\llbracket G' \rrbracket]$,
- $E'_i = \text{lift}_{W''', \Xi_i}(\rho[Y \mapsto \beta](G'))$,
- $E_i^* = \text{lift}_{W''', \Xi_i}(G_{ph})$, $G_{ph} \sqsubseteq \rho[Y \mapsto \beta](G_1)$,
- $\rho'' = \rho[Y \mapsto \beta][X \mapsto \alpha]$,
- $\varepsilon_{ih} = \langle E_i^*[\alpha^{E'_i}/X], E_i^*[E'_i/X] \rangle = \pi_2^*(\rho_i(\varepsilon^*))[\alpha, \rho(G'), Y][\hat{\beta}, \hat{\beta}]$ ($\varepsilon_{ih}.n = l$), such that

$$\varepsilon_{ih} \vdash W''', \Xi_i \vdash \rho[Y \mapsto \beta](G_1[\alpha/X]) \sim \rho[Y \mapsto \beta](G_1[G'/X])$$

$$(\downarrow_k W''', \varepsilon_{1h} v''_1 :: \rho[Y \mapsto \beta](G_1[G'/X]), \varepsilon_{2h} v''_2 :: \rho[Y \mapsto \beta](G_1[G'/X])) \in \mathcal{J}_{\rho[Y \mapsto \beta]}[\llbracket G_1[G'/X] \rrbracket]$$

If the combination of evidence does not succeed, then the result follows immediately. Otherwise, we get that

$$(\downarrow_{k+l} W''', W''', v'''_1, v'''_2) \in \mathcal{V}_{\rho[Y \mapsto \beta]}[\llbracket G_1[G'/X] \rrbracket]$$

where $v'''_i = (\varepsilon_i^* \circ (\varepsilon'_i \circ \rho'_i(\varepsilon^*))[\hat{G}_i^*, \hat{\beta}] \circ \varepsilon_{ih}) u'_i :: \rho[Y \mapsto \beta](G_1[G'/X])$

By the ascription Lemma A.152:

- $(\downarrow_{k+l}W''', v_1''', v_2''') \in \mathcal{V}_{\rho[Y \mapsto \beta]}[[G_1[G/X]]]$
- $\varepsilon'[Y] \Vdash \Xi; \Delta, Y \vdash G_1[G'/X] \sim G_1[G'/X] \ (\varepsilon'[Y].n = k')$
- $\downarrow_{k+l}W''' \in \mathcal{S}[[\Xi]]$ and $(\downarrow_{k+l}W''', \rho[Y \mapsto \beta]) \in \mathcal{D}[[\Delta, Y]]$

then we have:

$$\begin{aligned} & (\downarrow_{k+l}W''', \rho_1(\varepsilon'))[\hat{\beta}, \hat{\beta}]v_1''' :: \rho[Y \mapsto \beta](G_1[G'/X]), \\ & \rho_2(\varepsilon')[\hat{\beta}, \hat{\beta}]v_2''' :: \rho[Y \mapsto \beta](G_1[G'/X]) \in \mathcal{T}_{\rho[Y \mapsto \beta]}[[G_1[G/X]]] \end{aligned}$$

If the combination of evidence does not succeed, then the result follows immediately. Otherwise, we get that

$$(\downarrow_{k+l+k'}W''', v_1''', v_2''') \in \mathcal{V}_{\rho[Y \mapsto \beta]}[[G_1[G/X]]]$$

where $v_i'''' = (\varepsilon_i^* \circ (\varepsilon'_i \circ \rho'_i(\varepsilon^*)))[\hat{G}_i^*, \hat{\beta}] \circ \varepsilon_{ih} \circ \rho_i(\varepsilon')[\hat{\beta}, \hat{\beta}]u'_i :: \rho[Y \mapsto \beta](G_1[G'/X])$ Note that

$$W'''.\Xi_1 \triangleright (\varepsilon'_1 \circ \varepsilon_1 \circ \rho_1(\varepsilon'))[\hat{G}_1^*, \hat{\beta}]v'_1 :: \rho(G_1[G'/X][\beta/Y]) \xrightarrow{k+l+k'} W'''.\Xi_1 \triangleright v_1''''$$

And, we have to prove

$$\begin{aligned} & (W''', (\varepsilon'_1 \circ \varepsilon_1 \circ \rho_1(\varepsilon'))[\hat{G}_1^*, \hat{\beta}]v'_1 :: \rho(G_1[G'/X][\beta/Y]), (\varepsilon'_2 \circ \varepsilon_2 \circ \rho_2(\varepsilon'))[\hat{G}_2^*, \hat{\beta}]v'_2 :: \rho(G_1[G'/X][\beta/Y])) \\ & \in \mathcal{T}_{\rho[Y \mapsto \beta]}[[G_1[G'/X]]] \end{aligned}$$

Therefore, the result follows immediately $((\downarrow_{k+l+k'}W'''), v_1''', v_2''') \in \mathcal{V}_{\rho[Y \mapsto \beta]}[[G_1[G/X]]]$.

□

Lemma A.154 *If $\varepsilon \Vdash \Xi; \Delta \vdash \exists X.G_1 \sim \exists X.G_2$ then $\varepsilon[X] \Vdash \Xi; \Delta, X \vdash G_1 \sim G_2$.*

PROOF. Straightforward by induction on the evidences.

□

Lemma A.155

$$\varepsilon[E_1, E_2] = \pi_1^*(\varepsilon)[E_1, E_2] \circ \varepsilon[E_2, E_2] = \pi_1^2(\varepsilon)[E_1, E_2] \circ \varepsilon[E_2, E_2]$$

PROOF. Straightforward induction on the evidence structure.

□

Lemma A.156

$$(\varepsilon \circ \varepsilon')[E_1, E_2] = (\varepsilon \circ \pi_1^*(\varepsilon'))[E_1, E_2] \circ \varepsilon'[E_2, E_2] = (\varepsilon \circ \pi_1^2(\varepsilon'))[E_1, E_2] \circ \varepsilon'[E_2, E_2]$$

PROOF. Straightforward induction on the evidence structure.

□

Lemma A.157 *If $\varepsilon_i \Vdash W.\Xi_i \vdash \rho(G) \sim \rho(G)$, $W \in \mathcal{S}[[\Xi]]$ and $(W, \rho) \in \mathcal{D}[[\Delta]]$, then $\exists \varepsilon \Vdash \Xi, \Delta \vdash G \sim G$ such that $\varepsilon_i = \rho_i(\varepsilon)$.*

PROOF. Straightforward induction on the evidence structure.

□

Lemma A.158 (Evidence decomposition) *If*

- $\varepsilon_1 \Vdash \Xi; \Delta, X, Y \vdash G \sim G$
- $\varepsilon_2 \Vdash \Xi; \Delta, X \vdash G[G'/Y] \sim G''$ and $\Xi; \Delta \vdash G'$
- $W \in \mathcal{S}[\Xi], (W, \rho[X \mapsto \alpha][Y \mapsto \beta]) \in \mathcal{D}[\Delta, X, Y], W.\Xi_i(\alpha) = \rho(G_i)$ and $W.\Xi_i(\beta) = \rho(G')$

then $\exists \varepsilon \Vdash \Xi; \Delta, X, Y \vdash G \sim G$

$$(\rho_i(\varepsilon_1)[\beta, G', Y] \sharp \rho_i(\varepsilon_2))[G_i, \alpha, X] = (\rho_i(\varepsilon)[\beta, \beta, Y])[G_i, \alpha, X] \sharp (\pi_2^*(\rho_i(\varepsilon))[\beta, G, Y] \sharp \rho_i(\varepsilon_2))[\alpha, \alpha, X]$$

PROOF. We proceed by induction on G .

Case ($G = B$ and $G'' = B$). Then, we know that $\varepsilon_i = \langle B, B \rangle$. Therefore, if we choose $\varepsilon = \langle B, B \rangle$ the results follows immediately.

Case ($G = G'_1 \rightarrow G'_2$, and $G'' = G'_1 \rightarrow G'_2$). We know that

- $\varepsilon_1 \Vdash \Xi; \Delta, X, Y \vdash G'_1 \rightarrow G'_2 \sim G'_1 \rightarrow G'_2$ implies that

$$\text{idom}^\sharp(\varepsilon_1) \Vdash \Xi; \Delta, X, Y \vdash G'_1 \sim G'_1$$
- $\varepsilon_2 \Vdash \Xi; \Delta, X \vdash (G'_1 \rightarrow G'_2)[G'/Y] \sim G'_1 \rightarrow G'_2$ implies that

$$\text{idom}^\sharp(\varepsilon_2) \Vdash \Xi; \Delta, X \vdash G'_1[G'/Y] \sim G'_1$$

Therefore by the induction hypothesis, we know that $\exists \varepsilon' \Vdash \Xi; \Delta, X, Y \vdash G'_1 \sim G'_1$ such that

$$\begin{aligned} & (\rho_i(\text{idom}^\sharp(\varepsilon_1))[\beta, G', Y] \sharp \rho_i(\text{idom}^\sharp(\varepsilon_2)))[G_i, \alpha, X] = \\ & (\rho_i(\varepsilon')[\beta, \beta, Y])[G_i, \alpha, X] \sharp (\pi_2^*(\rho_i(\varepsilon'))[\beta, G, Y] \sharp \rho_i(\text{idom}^\sharp(\varepsilon_2)))[\alpha, \alpha, X] \end{aligned}$$

Also we know that

- $\varepsilon_1 \Vdash \Xi; \Delta, X, Y \vdash G'_1 \rightarrow G'_2 \sim G'_1 \rightarrow G'_2$ implies that

$$\text{icod}^\sharp(\varepsilon_1) \Vdash \Xi; \Delta, X, Y \vdash G'_2 \sim G'_2$$
- $\varepsilon_2 \Vdash \Xi; \Delta, X \vdash (G'_1 \rightarrow G'_2)[G'/Y] \sim G'_1 \rightarrow G'_2$ implies that

$$\text{icod}^\sharp(\varepsilon_2) \Vdash \Xi; \Delta, X \vdash G'_2[G'/Y] \sim G'_2$$

Therefore by the induction hypothesis, we know that $\exists \varepsilon'' \Vdash \Xi; \Delta, X, Y \vdash G'_2 \sim G'_2$ such that

$$\begin{aligned} & (\rho_i(\text{icod}^\sharp(\varepsilon_1))[\beta, G', Y] \sharp \rho_i(\text{icod}^\sharp(\varepsilon_2)))[G_i, \alpha, X] = \\ & (\rho_i(\varepsilon'')[\beta, \beta, Y])[G_i, \alpha, X] \sharp (\pi_2^*(\rho_i(\varepsilon''))[\beta, G, Y] \sharp \rho_i(\text{icod}^\sharp(\varepsilon_2)))[\alpha, \alpha, X] \end{aligned}$$

Therefore, it follows that $\exists \varepsilon \Vdash \Xi; \Delta, X, Y \vdash G'_1 \rightarrow G'_2 \sim G'_1 \rightarrow G'_2$, such that the result follows immediately ($\varepsilon = \langle \pi_1(\varepsilon') \rightarrow \pi_1(\varepsilon''), \pi_2(\varepsilon') \rightarrow \pi_2(\varepsilon'') \rangle$). Note that

- $idom^\sharp(\varepsilon) = \varepsilon'$
- $icod^\sharp(\varepsilon) = \varepsilon''$
- $idom^\sharp((\rho_i(\varepsilon_1)[\beta, G', Y] \circledast \rho_i(\varepsilon_2))[G_i, \alpha, X]) =$
 $(\rho_i(idom^\sharp(\varepsilon_1))[\beta, G', Y] \circledast \rho_i(idom^\sharp(\varepsilon_2)))[G_i, \alpha, X] =$
 $(\rho_i(\varepsilon')[\beta, \beta, Y])[G_i, \alpha, X] \circledast (\pi_2^*(\rho_i(\varepsilon'))[\beta, G, Y] \circledast \rho_i(idom^\sharp(\varepsilon_2)))[\alpha, \alpha, X] =$
 $idom^\sharp((\rho_i(\varepsilon)[\beta, \beta, Y])[G_i, \alpha, X] \circledast (\pi_2^*(\rho_i(\varepsilon))[\beta, G, Y] \circledast \rho_i(\varepsilon_2)))[\alpha, \alpha, X]$
- $icod^\sharp((\rho_i(\varepsilon_1)[\beta, G', Y] \circledast \rho_i(\varepsilon_2))[G_i, \alpha, X]) =$
 $(icod^\sharp(\rho_i(icod^\sharp(\varepsilon_1))[\beta, G', Y] \circledast \rho_i(icod^\sharp(\varepsilon_2)))[G_i, \alpha, X] =$
 $(\rho_i(\varepsilon''[\beta, \beta, Y])[G_i, \alpha, X] \circledast (\pi_2^*(\rho_i(\varepsilon''))[\beta, G, Y] \circledast \rho_i(icod^\sharp(\varepsilon_2)))[\alpha, \alpha, X] =$
 $icod^\sharp((\rho_i(\varepsilon)[\beta, \beta, Y])[G_i, \alpha, X] \circledast (\pi_2^*(\rho_i(\varepsilon))[\beta, G, Y] \circledast \rho_i(\varepsilon_2)))[\alpha, \alpha, X]$

Note that two evidences are equals if and only if their $idom^\sharp$ and $icod^\sharp$ equals too.

Case ($G = \forall X.G'_1$ and $G'' = \forall X.G'_1$). Similar to function case.

Case ($G = G_1 \times G_2$). Similar to function case.

Case ($G = \alpha$). This means that evidences do not have type variables, therefore, type substitutions are not applied. For this reason, the result follows immediately.

Case ($G = \beta$). This means that evidences do not have type variables, therefore, type substitutions are not applied. For this reason, the result follows immediately.

Case ($G = \beta$). This means that evidences do not have type variables, therefore, type substitutions are not applied. For this reason, the result follows immediately.

Case ($G = X$). Then, we know that $\varepsilon_1 = \langle X, X \rangle$ and $\varepsilon_2 = \langle X, X \rangle$. Therefore, with $\varepsilon = \langle X, X \rangle$ the result follows immediately.

Case ($G = Y$). Then, we know that $\varepsilon_1 = \langle Y, Y \rangle$. Since, $\varepsilon_2 \Vdash \Xi; \Delta, X \vdash G' \sim G''$ and $\Xi; \Delta \vdash G'$ (without X), we know that

$$\rho_i(\varepsilon_2)[G_i, \alpha, X] = \rho_i(\varepsilon_2)[\alpha, \alpha, X] = \rho_i(\varepsilon_2)$$

Therefore, $\exists \varepsilon = \langle Y, Y \rangle$, such that the result follows immediately.

Case ($G = Z$). Then, we know that $\varepsilon_1 = \langle Z, Z \rangle$ and $\varepsilon_2 = \langle Z, Z \rangle$. Therefore, with $\varepsilon = \langle Z, Z \rangle$ the result follows immediately.

Case ($G = ?$). We follow by case in the evidences.

- $\varepsilon_1 = \langle ?, ? \rangle$, then $\exists \varepsilon = \varepsilon_2$ such that the results follows immediately (by Lemma A.155).
- $\varepsilon_2 = \langle ?, ? \rangle$, then $\exists \varepsilon = \varepsilon_1$ such that the results follows immediately (by Lemma A.155).
- The other evidence cases are covered in other cases of the proof.

□

Proposition A.159 *If* $\Xi; \Delta; \Gamma \vdash t_1 \approx t_2 : G$, *then* $\Xi; \Delta; \Gamma \vdash t_1 \approx^{ctx} t_2 : G$.

PROOF. Similar to Th. A.117.

□

A.10.5 A Weak Dynamic Gradual Guarantee for GSF[□]

Proposition A.160 *If $\Omega \vdash t_1^* : G_1^* \leq t_2^* : G_2^*$, $\Omega \equiv \Gamma_1 \sqsubseteq \Gamma_2$, $\Delta; \Gamma_i \vdash t_i^* \rightsquigarrow t_i^{**} : G_i^*$, then $\Omega \vdash \triangleright t_1^{**} : G_1^* \leq \Xi_2 \triangleright t_2^{**} : G_2^*$.*

PROOF. We follow by induction on $\Omega \vdash t_1^* : G_1^* \leq t_2^* : G_2^*$. We avoid the notation $\Omega \vdash t_1^* : G_1^* \leq t_2^* : G_2^*$, and use $t_1^* \leq t_2^*$ instead, for simplicity, when the typing environments are not relevant. We use metavariable v or u in GSF to range over constants, functions and type abstractions. We only proof here the cases related to existential types. Other cases were proved in Section A.5.

Case ($\leq v$).

$$\begin{aligned} & (\leq v) \frac{\Omega \vdash u_1 : G_1^* \leq_v u_2 : G_2^* \quad G_1^* \leq G_2^*}{\Omega \vdash u_1 : G_1^* \leq u_2 : G_2^*} \\ & (\text{Gu}) \frac{\Delta; \Gamma_1 \vdash u_1 \rightsquigarrow u'_1 : G_1^* \quad \varepsilon_{G_1^*} = \mathcal{J}(\cdot, \cdot)G_1^*, G_1^*}{\Delta; \Gamma_1 \vdash u_1 \rightsquigarrow \varepsilon_{G_1^*} u'_1 :: G_1^* : G_1^*} \\ & (\text{Gu}) \frac{\Delta; \Gamma_2 \vdash u_2 \rightsquigarrow u'_2 : G_2^* \quad \varepsilon_{G_2^*} = \mathcal{J}(\cdot, \cdot)G_2^*, G_2^*}{\Delta; \Gamma_2 \vdash u_2 \rightsquigarrow \varepsilon_{G_2^*} u'_2 :: G_2^* : G_2^*} \end{aligned}$$

We have to prove that $\Omega \vdash \Xi_1 \triangleright \varepsilon_{G_1^*} u'_1 :: G_1^* : G_1^* \leq \Xi_2 \triangleright \varepsilon_{G_2^*} u'_2 :: G_2^* : G_2^*$. By the rule ($\leq \text{asc}_\varepsilon$), we are required to prove that $\varepsilon_{G_1^*} \leq \varepsilon_{G_2^*}$, $\Omega \vdash \Xi_1 \triangleright u'_1 : G_1^* \leq \Xi_2 \triangleright u'_2 : G_2^*$ and $G_1^* \sqsubseteq G_2^*$. Since $G_1^* \leq G_2^*$ and Proposition A.170, we know that $\varepsilon_{G_1^*} \leq \varepsilon_{G_2^*}$. Also, by Proposition A.171 and $G_1^* \leq G_2^*$, we now that $G_1^* \sqsubseteq G_2^*$. Therefore, we only have required to prove that $\Omega \vdash \Xi_1 \triangleright u'_1 : G_1^* \leq \Xi_2 \triangleright u'_2 : G_2^*$. We follow by case analysis on $\Omega \vdash u_1 : G_1^* \leq_v u_2 : G_2^*$. We only take into account the package, where $u_i = \text{packu}\langle G'_i, v_i \rangle$ as $\exists X.G''_i$ and $G_i^* = \exists X.G''_i$, where $\exists X.G''_1 \leq \exists X.G''_2$. We know that

$$\begin{aligned} & (\leq \text{pack}) \frac{G_1^* \leq G_2^* \quad \Omega \vdash \triangleright v_1 : G_1 \leq \Xi_2 \triangleright v_2 : G_2 \quad \exists X.G''_1 \sqsubseteq \exists X.G''_2 \quad G_1 \text{pullback} G''_1[G'_1/X] \leq G_2 \text{pullback} G''_2[G'_2/X]}{\Omega \vdash \triangleright \text{pack}\langle G'_1, v_1 \rangle \text{ as } \exists X.G''_1 : \exists X.G''_1 \leq \Xi_2 \triangleright \text{pack}\langle G'_2, v_2 \rangle \text{ as } \exists X.G''_2 : \exists X.G''_2} \\ & (\text{Gpack}) \frac{\Delta; \Gamma_1 \vdash v_1 :: G''_1[G'_1/X] \rightsquigarrow v''_1 : G''_1[G'_1/X]}{\Delta; \Gamma_1 \vdash \text{pack}\langle G'_1, v_1 \rangle \text{ as } \exists X.G''_1 \rightsquigarrow \text{packu}\langle G'_1, v''_1 \rangle \text{ as } \exists X.G''_1 : \exists X.G''_1} \\ & (\text{Gpack}) \frac{\Delta; \Gamma_2 \vdash v_2 :: G''_2[G'_2/X] \rightsquigarrow v''_2 : G''_2[G'_2/X]}{\Delta; \Gamma_2 \vdash \text{pack}\langle G'_2, v_2 \rangle \text{ as } \exists X.G''_2 \rightsquigarrow \text{packu}\langle G'_2, v''_2 \rangle \text{ as } \exists X.G''_2 : \exists X.G''_2} \end{aligned}$$

We have to prove that $\Omega \vdash \triangleright \text{packu}\langle G'_1, v''_1 \rangle \text{ as } \exists X.G''_1 : \exists X.G''_1 \leq \Xi_2 \triangleright \text{packu}\langle G'_2, v''_2 \rangle \text{ as } \exists X.G''_2 : \exists X.G''_2$, or what is the same by the rule ($\leq \text{packu}_\varepsilon$), we have to prove that $G'_1 \leq G'_2$, $\Omega \vdash \triangleright v''_1 : G''_1[G'_1/X] \leq \Xi_2 \triangleright v''_2 : G''_2[G'_2/X]$ and $\exists X.G''_1 \sqsubseteq \exists X.G''_2$. By premise, $G'_1 \leq G'_2$ and $\exists X.G''_1 \sqsubseteq \exists X.G''_2$ (Proposition A.164) follows immediately. Therefore, we only have required to prove that $\Omega \vdash \triangleright v''_1 : G''_1[G'_1/X] \leq \Xi_2 \triangleright v''_2 : G''_2[G'_2/X]$, which follows by the induction hypothesis. We know that

$$v''_1 = \varepsilon_1 v'_1 :: G''_1[G'_1/X] \text{ where } \varepsilon_1 = \mathcal{J}(\cdot, \cdot)G_1, G''_1[G'_1/X]$$

$$v''_2 = \varepsilon_2 v'_2 :: G''_2[G'_2/X] \text{ where } \varepsilon_2 = \mathcal{J}(\cdot, \cdot)G_2, G''_2[G'_2/X]$$

where $\Delta; \Gamma_i \vdash v_i \rightsquigarrow_v v'_i : G_i$, and therefore $\Omega \vdash \Xi_1 \triangleright v'_1 : G_1 \leq \Xi_2 \triangleright v'_2 : G_2$.

By rule ($\leq\text{asc}_\varepsilon$), we are required to prove that $\varepsilon_1 \leq \varepsilon_2$, $\Omega \vdash \Xi_1 \triangleright v'_1 : G_1 \leq \Xi_2 \triangleright v'_2 : G_2$ and $G'_1[G'_1/X] \sqsubseteq G'_2[G'_2/X]$. By induction hypothesis on $\Omega \vdash \Xi_1 \triangleright v_1 : G_1 \leq \Xi_2 \triangleright v_2 : G_2$, we know that $\Omega \vdash \Xi_1 \triangleright v'_1 : G_1 \leq \Xi_2 \triangleright v'_2 : G_2$. By Proposition A.174, $G'_1 \leq G'_2$ and $G'_1 \leq G'_2$, we know that $G'_1[G'_1/X] \leq G'_2[G'_2/X]$, and therefore $G''_1[G'_1/X] \sqsubseteq G''_2[G'_2/X]$. By Proposition A.162 and $G_1\text{pullback}G''_1[G'_1/X] \leq G_2\text{pullback}G''_2[G'_2/X]$, we know that

$$\begin{aligned} \varepsilon_1 &= \mathcal{I}(\cdot, \cdot)G_1, G''_1[G'_1/X] = \mathcal{I}(\cdot, \cdot)G_1\text{pullback}G''_1[G'_1/X], G_1\text{pullback}G''_1[G'_1/X] \leq \\ &\mathcal{I}(\cdot, \cdot)G_2\text{pullback}G''_2[G'_2/X], G_2\text{pullback}G''_2[G'_2/X] = \mathcal{I}(\cdot, \cdot)G_2, G''_2[G'_2/X] = \varepsilon_2 \end{aligned}$$

Therefore, the results holds.

Case ($\leq\text{ascv}$). We know that

$$\begin{aligned} (\leq\text{ascv}) \frac{\Omega \vdash u_1 : G_1^{**} \leq_v u_2 : G_2^{**} \quad G_1^{**}\text{pullback}G_1^* \leq G_2^{**}\text{pullback}G_2^* \quad G_1^* \sqsubseteq G_2^*}{\Omega \vdash u_1 :: G_1^* : G_1^* \leq u_2 :: G_2^* : G_2^*} \\ (\text{Gascu}) \frac{\Delta; \Gamma_1 \vdash u_1 \rightsquigarrow u'_1 : G_1^{**} \quad \varepsilon_1 = \mathcal{I}(\cdot, \cdot)G_1^{**}, G_1^*}{\Delta; \Gamma_1 \vdash u_1 :: G_1^* \rightsquigarrow \varepsilon_1 u'_1 :: G_1^* : G_1^*} \\ (\text{Gascu}) \frac{\Delta; \Gamma_2 \vdash u_2 \rightsquigarrow u'_2 : G_2^{**} \quad \varepsilon_2 = \mathcal{I}(\cdot, \cdot)G_2^{**}, G_2^*}{\Delta; \Gamma_2 \vdash u_2 :: G_2^* \rightsquigarrow \varepsilon_2 u'_2 :: G_2^* : G_2^*} \end{aligned}$$

We have to prove that $\Omega \vdash \Xi_1 \triangleright \varepsilon_1 u'_1 :: G_1^* : G_1^* \leq \Xi_2 \triangleright \varepsilon_2 u'_2 :: G_2^* : G_2^*$, or what is the same by the rule ($\leq\text{asc}_\varepsilon$), we have to prove that $\varepsilon_1 \leq \varepsilon_2$, $\Omega \vdash \Xi_1 \triangleright u'_1 : G_1^{**} \leq \Xi_2 \triangleright u'_2 : G_2^{**}$ and $G_1^* \sqsubseteq G_2^*$. By Proposition A.161, we know that $\varepsilon_1 = \mathcal{I}(\cdot, \cdot)G_1^{**}, G_1^* = \mathcal{I}(\cdot, \cdot)G_1^{**}\text{pullback}G_1^*, G_1^{**}\text{pullback}G_1^*$ and $\varepsilon_2 = \mathcal{I}(\cdot, \cdot)G_2^{**}, G_2^* = \mathcal{I}(\cdot, \cdot)G_2^{**}\text{pullback}G_2^*, G_2^{**}\text{pullback}G_2^*$. Since $G_1^{**}\text{pullback}G_1^* \leq G_2^{**}\text{pullback}G_2^*$, then $\varepsilon_1 = \mathcal{I}(\cdot, \cdot)G_1^{**}, G_1^* = \mathcal{I}(\cdot, \cdot)G_1^{**}\text{pullback}G_1^*, G_1^{**}\text{pullback}G_1^* \leq \mathcal{I}(\cdot, \cdot)G_2^{**}\text{pullback}G_2^*, G_2^{**}\text{pullback}G_2^* = \mathcal{I}(\cdot, \cdot)G_2^{**}, G_2^* = \varepsilon_2$, by Proposition A.162. Thus, we only have to prove that $\Omega \vdash \Xi_1 \triangleright u'_1 : G_1^{**} \leq \Xi_2 \triangleright u'_2 : G_2^{**}$, and we know that $\Omega \vdash u'_1 : G_1^{**} \leq_v u'_2 : G_2^{**}$. We follow by case analysis on $\Omega \vdash u_1 : G_1^{**} \leq_v u_2 : G_2^{**}$. We only take into account the package, where $u_i = \text{pack}\langle G'_i, v_i \rangle$ as $\exists X.G'_i$ and $G_i^* = \exists X.G''_i$, where $\exists X.G'_i \leq \exists X.G''_i$. We know that

$$\begin{aligned} (\leq\text{pack}) \frac{G'_1 \leq G'_2 \quad \Omega \vdash \triangleright v_1 : G_1 \leq \Xi_2 \triangleright v_2 : G_2 \quad \exists X.G''_1 \sqsubseteq \exists X.G''_2 \quad G_1\text{pullback}G''_1[G'_1/X] \leq G_2\text{pullback}G''_2[G'_2/X]}{\Omega \vdash \triangleright \text{pack}\langle G'_1, v_1 \rangle \text{ as } \exists X.G''_1 : \exists X.G''_1 \leq \Xi_2 \triangleright \text{pack}\langle G'_2, v_2 \rangle \text{ as } \exists X.G''_2 : \exists X.G''_2} \\ (\text{Gpack}) \frac{\Delta; \Gamma_1 \vdash v_1 :: G''_1[G'_1/X] \rightsquigarrow v''_1 : G''_1[G'_1/X]}{\Delta; \Gamma_1 \vdash \text{pack}\langle G'_1, v_1 \rangle \text{ as } \exists X.G''_1 \rightsquigarrow \text{packu}\langle G'_1, v''_1 \rangle \text{ as } \exists X.G''_1 : \exists X.G''_1} \\ (\text{Gpack}) \frac{\Delta; \Gamma_2 \vdash v_2 :: G''_2[G'_2/X] \rightsquigarrow v''_2 : G''_2[G'_2/X]}{\Delta; \Gamma_2 \vdash \text{pack}\langle G'_2, v_2 \rangle \text{ as } \exists X.G''_2 \rightsquigarrow \text{packu}\langle G'_2, v''_2 \rangle \text{ as } \exists X.G''_2 : \exists X.G''_2} \end{aligned}$$

We have to prove that $\Omega \vdash \triangleright \text{packu}\langle G'_1, v''_1 \rangle \text{ as } \exists X.G''_1 : \exists X.G''_1 \leq \Xi_2 \triangleright \text{packu}\langle G'_2, v''_2 \rangle \text{ as } \exists X.G''_2 : \exists X.G''_2$, or what is the same by the rule ($\leq\text{packu}_\varepsilon$), we have to prove that $G'_1 \leq G'_2$, $\Omega \vdash \triangleright v''_1 : G''_1[G'_1/X] \leq \Xi_2 \triangleright v''_2 : G''_2[G'_2/X]$ and $\exists X.G''_1 \sqsubseteq \exists X.G''_2$. By premise, $G'_1 \leq G'_2$ and $\exists X.G''_1 \sqsubseteq \exists X.G''_2$ (Proposition A.164) follows immediately. Therefore, we only have required to prove that $\Omega \vdash \triangleright v''_1 : G''_1[G'_1/X] \leq \Xi_2 \triangleright v''_2 : G''_2[G'_2/X]$, which follows by the induction hypothesis.

We know that

$$v_1'' = \varepsilon_1 v_1' :: G_1''[G_1'/X] \text{ where } \varepsilon_1 = \mathcal{I}(\cdot, \cdot)G_1, G_1''[G_1'/X]$$

$$v_2'' = \varepsilon_2 v_2' :: G_2''[G_2'/X] \text{ where } \varepsilon_2 = \mathcal{I}(\cdot, \cdot)G_2, G_2''[G_2'/X]$$

where $\Delta; \Gamma_i \vdash v_i \rightsquigarrow_v v_i' : G_i$, and therefore $\Omega \vdash \Xi_1 \triangleright v_1' : G_1 \leq \Xi_2 \triangleright v_2' : G_2$.

By rule ($\leq\text{asc}_\varepsilon$), we are required to prove that $\varepsilon_1 \leq \varepsilon_2$, $\Omega \vdash \Xi_1 \triangleright v_1' : G_1 \leq \Xi_2 \triangleright v_2' : G_2$ and $G_1''[G_1'/X] \sqsubseteq G_2''[G_2'/X]$. By induction hypothesis on $\Omega \vdash \Xi_1 \triangleright v_1 : G_1 \leq \Xi_2 \triangleright v_2 : G_2$, we know that $\Omega \vdash \Xi_1 \triangleright v_1' : G_1 \leq \Xi_2 \triangleright v_2' : G_2$. By Proposition A.174, $G_1'' \leq G_2''$ and $G_1' \leq G_2'$, we know that $G_1''[G_1'/X] \leq G_2''[G_2'/X]$, and therefore $G_1''[G_1'/X] \sqsubseteq G_2''[G_2'/X]$. By Proposition A.162 and $G_1\text{pullback}G_1''[G_1'/X] \leq G_2\text{pullback}G_2''[G_2'/X]$, we know that

$$\begin{aligned} \varepsilon_1 &= \mathcal{I}(\cdot, \cdot)G_1, G_1''[G_1'/X] = \mathcal{I}(\cdot, \cdot)G_1\text{pullback}G_1''[G_1'/X], G_1\text{pullback}G_1''[G_1'/X] \leq \\ &\mathcal{I}(\cdot, \cdot)G_2\text{pullback}G_2''[G_2'/X], G_2\text{pullback}G_2''[G_2'/X] = \mathcal{I}(\cdot, \cdot)G_2, G_2''[G_2'/X] = \varepsilon_2 \end{aligned}$$

Therefore, the results holds.

Case ($\leq\text{pack}$). We know that

$$(\leq\text{pack}) \frac{G_1' \leq G_2' \quad \Omega \vdash \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2 \quad \exists X.G_1'' \leq \exists X.G_2'' \quad G_1\text{pullback}G_1''[G_1'/X] \leq G_2\text{pullback}G_2''[G_2'/X]}{\Omega \vdash \triangleright \text{pack}\langle G_1', t_1 \rangle \text{ as } \exists X.G_1'' : \exists X.G_1'' \leq \Xi_2 \triangleright \text{pack}\langle G_2', t_2 \rangle \text{ as } \exists X.G_2'' : \exists X.G_2''}$$

$$(\text{Gpack}) \frac{\Delta; \Gamma_1 \vdash t_1 \rightsquigarrow t_1' : G_1 \quad t_1'' = \text{norm}(t_1', G_1, G_1''[G_1'/X])}{\Delta; \Gamma_1 \vdash \text{pack}\langle G_1', t_1 \rangle \text{ as } \exists X.G_1'' \rightsquigarrow \text{pack}\langle G_1', t_1'' \rangle \text{ as } \exists X.G_1'' : \exists X.G_1''}$$

$$(\text{Gpack}) \frac{\Delta; \Gamma_2 \vdash t_2 \rightsquigarrow t_2' : G_2 \quad t_2'' = \text{norm}(t_2', G_2, G_2''[G_2'/X])}{\Delta; \Gamma_2 \vdash \text{pack}\langle G_2', t_2 \rangle \text{ as } \exists X.G_2'' \rightsquigarrow \text{pack}\langle G_2', t_2'' \rangle \text{ as } \exists X.G_2'' : \exists X.G_2''}$$

We have to prove that $\Omega \vdash \triangleright \text{pack}\langle G_1', t_1'' \rangle \text{ as } \exists X.G_1'' : \exists X.G_1'' \leq \Xi_2 \triangleright \text{pack}\langle G_2', t_2'' \rangle \text{ as } \exists X.G_2'' : \exists X.G_2''$, or what is the same by the rule ($\leq\text{pack}_\varepsilon$), we have to prove that $G_1' \leq G_2'$, $\Omega \vdash \triangleright t_1'' : G_1''[G_1'/X] \leq \Xi_2 \triangleright t_2'' : G_2''[G_2'/X]$ and $\exists X.G_1'' \leq \exists X.G_2''$. By premise, $G_1' \leq G_2'$ and $\exists X.G_1'' \leq \exists X.G_2''$ (Proposition A.164) follows immediately. Therefore, we only have required to prove that $\Omega \vdash \triangleright t_1'' : G_1''[G_1'/X] \leq \Xi_2 \triangleright t_2'' : G_2''[G_2'/X]$. We know that

$$t_1'' = \text{norm}(t_1', G_1, G_1''[G_1'/X]) = \varepsilon_1 t_1' :: G_1''[G_1'/X] \text{ where } \varepsilon_1 = \mathcal{I}(\cdot, \cdot)G_1, G_1''[G_1'/X]$$

$$t_2'' = \text{norm}(t_2', G_2, G_2''[G_2'/X]) = \varepsilon_2 t_2' :: G_2''[G_2'/X] \text{ where } \varepsilon_2 = \mathcal{I}(\cdot, \cdot)G_2, G_2''[G_2'/X]$$

By rule ($\leq\text{asc}_\varepsilon$), we are required to prove that $\varepsilon_1 \leq \varepsilon_2$, $\Omega \vdash \Xi_1 \triangleright t_1' : G_1 \leq \Xi_2 \triangleright t_2' : G_2$ and $G_1''[G_1'/X] \sqsubseteq G_2''[G_2'/X]$. By induction hypothesis on $\Omega \vdash \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$, we know that $\Omega \vdash \Xi_1 \triangleright t_1' : G_1 \leq \Xi_2 \triangleright t_2' : G_2$. By Proposition A.174, $G_1'' \leq G_2''$ and $G_1' \leq G_2'$, we know that $G_1''[G_1'/X] \leq G_2''[G_2'/X]$, and therefore $G_1''[G_1'/X] \sqsubseteq G_2''[G_2'/X]$. By Proposition A.162 and $G_1\text{pullback}G_1''[G_1'/X] \leq G_2\text{pullback}G_2''[G_2'/X]$, we know that

$$\begin{aligned} \varepsilon_1 &= \mathcal{I}(\cdot, \cdot)G_1, G_1''[G_1'/X] = \mathcal{I}(\cdot, \cdot)G_1\text{pullback}G_1''[G_1'/X], G_1\text{pullback}G_1''[G_1'/X] \leq \\ &\mathcal{I}(\cdot, \cdot)G_2\text{pullback}G_2''[G_2'/X], G_2\text{pullback}G_2''[G_2'/X] = \mathcal{I}(\cdot, \cdot)G_2, G_2''[G_2'/X] = \varepsilon_2 \end{aligned}$$

Therefore, the results holds.

Case (unpack). We know that

$$(\leq\text{unpack}) \frac{\Omega \vdash \triangleright t_{11} : G_1 \leq \Xi_2 \triangleright t_{21} : G_2 \quad \Omega, x : \text{schm}_e^\sharp(G_1) \sqsubseteq \text{schm}_e^\sharp(G_2) \vdash \triangleright t_{12} : G'_1 \leq \Xi_2 \triangleright t_{22} : G'_2}{\Omega \vdash \triangleright \text{unpack}\langle X, x \rangle = t_{11} \text{ in } t_{12} : G'_1 \leq \Xi_2 \triangleright \text{unpack}\langle X, x \rangle = t_{21} \text{ in } t_{22} : G'_2}$$

$$(\text{Gunpack}) \frac{\Delta; \Gamma_1 \vdash t_{11} \rightsquigarrow t'_{11} : G_1 \quad t''_{11} = \text{norm}(t'_{11}, G_1, \exists \text{var}^\sharp(G_1). \text{schm}_e^\sharp(G_1)) \quad \Delta; \Gamma_1, x : \text{schm}_e^\sharp(G_1) \vdash t_{12} \rightsquigarrow t'_{12} : G'_1}{\Delta; \Gamma_1 \vdash \text{unpack}\langle X, x \rangle = t_{11} \text{ in } t_{12} \rightsquigarrow \text{unpack}\langle X, x \rangle = t''_{11} \text{ in } t'_{12} : G'_1}$$

$$(\text{Gunpack}) \frac{\Delta; \Gamma_2 \vdash t_{21} \rightsquigarrow t'_{21} : G_2 \quad t''_{21} = \text{norm}(t'_{21}, G_2, \exists \text{var}^\sharp(G_2). \text{schm}_e^\sharp(G_2)) \quad \Delta; \Gamma_2, x : \text{schm}_e^\sharp(G_2) \vdash t_{22} \rightsquigarrow t'_{22} : G'_2}{\Delta; \Gamma_2 \vdash \text{unpack}\langle X, x \rangle = t_{21} \text{ in } t_{22} \rightsquigarrow \text{unpack}\langle X, x \rangle = t''_{21} \text{ in } t'_{22} : G'_2}$$

We have to prove that $\Omega \vdash \Xi_1 \triangleright \text{unpack}\langle X, x \rangle = t''_{11} \text{ in } t'_{12} : G'_1 \leq \Xi_2 \triangleright \text{unpack}\langle X, x \rangle = t''_{21} \text{ in } t'_{22} : G'_2$, or what is the same by the rule ($\leq\text{unpack}_\varepsilon$), we have to prove that $\Omega \vdash \Xi_1 \triangleright t''_{11} : \exists \text{var}^\sharp(G_1). \text{schm}_e^\sharp(G_1) \leq \Xi_2 \triangleright t''_{21} : \exists \text{var}^\sharp(G_2). \text{schm}_e^\sharp(G_2)$ and $\Omega, x : \text{schm}_e^\sharp(G_1) \sqsubseteq \text{schm}_e^\sharp(G_2) \vdash \Xi_1 \triangleright t'_{12} : G'_1 \leq \Xi_2 \triangleright t'_{22} : G'_2$. By the induction hypothesis on $\Omega, x : \text{schm}_e^\sharp(G_1) \sqsubseteq \text{schm}_e^\sharp(G_2) \vdash \triangleright t_{12} : G'_1 \leq \Xi_2 \triangleright t_{22} : G'_2$, we know that $\Omega, x : \text{schm}_e^\sharp(G_1) \sqsubseteq \text{schm}_e^\sharp(G_2) \vdash \Xi_1 \triangleright t'_{12} : G'_1 \leq \Xi_2 \triangleright t'_{22} : G'_2$. Therefore, we only are required to prove that $\Omega \vdash \Xi_1 \triangleright t''_{11} : \exists \text{var}^\sharp(G_1). \text{schm}_e^\sharp(G_1) \leq \Xi_2 \triangleright t''_{21} : \exists \text{var}^\sharp(G_2). \text{schm}_e^\sharp(G_2)$. We know that

$$t''_{11} = \text{norm}(t'_{11}, G_1, \exists \text{var}^\sharp(G_1). \text{schm}_e^\sharp(G_1)) = \varepsilon_1 t'_{11} :: \exists \text{var}^\sharp(G_1). \text{schm}_e^\sharp(G_1)$$

where $\varepsilon_1 = \mathcal{I}(\cdot, ()G_1, \exists \text{var}^\sharp(G_1). \text{schm}_e^\sharp(G_1)) = \mathcal{I}(\cdot, ()\exists \text{var}^\sharp(G_1). \text{schm}_e^\sharp(G_1), \exists \text{var}^\sharp(G_1). \text{schm}_e^\sharp(G_1)) = \varepsilon_{\exists \text{var}^\sharp(G_1). \text{schm}_e^\sharp(G_1)}$

$$t''_{21} = \text{norm}(t'_{21}, G_2, \exists \text{var}^\sharp(G_2). \text{schm}_e^\sharp(G_2)) = \varepsilon_2 t'_{21} :: \exists \text{var}^\sharp(G_2). \text{schm}_e^\sharp(G_2)$$

where $\varepsilon_2 = \mathcal{I}(\cdot, ()G_2, \exists \text{var}^\sharp(G_2). \text{schm}_e^\sharp(G_2)) = \mathcal{I}(\cdot, ()\exists \text{var}^\sharp(G_2). \text{schm}_e^\sharp(G_2), \exists \text{var}^\sharp(G_2). \text{schm}_e^\sharp(G_2)) = \varepsilon_{\exists \text{var}^\sharp(G_2). \text{schm}_e^\sharp(G_2)}$

By induction hypothesis on $\Omega \vdash t_{11} : G_1 \leq t_{21} : G_2$, we know that $\Omega \vdash \triangleright t'_{11} : G_1 \leq \Xi_2 \triangleright t'_{21} : G_2$, and by Proposition A.164, we know that $G_1 \sqsubseteq G_2$, thus $\exists \text{var}^\sharp(G_1). \text{schm}_e^\sharp(G_1) \sqsubseteq \exists \text{var}^\sharp(G_2). \text{schm}_e^\sharp(G_2)$. Therefore, we only have to prove by rule ($\leq\text{Masc}_\varepsilon$) that $\varepsilon_1 \sqsubseteq \varepsilon_2$. But, by Proposition A.163 and $\exists \text{var}^\sharp(G_1). \text{schm}_e^\sharp(G_1) \sqsubseteq \exists \text{var}^\sharp(G_2). \text{schm}_e^\sharp(G_2)$ the results holds. □

Proposition A.161 $\mathcal{I}(\Xi, ()G_1 \text{pullback } G_2, G_1 \text{pullback } G_2) = \mathcal{I}(\Xi, ()G_1, G_2)$

PROOF. By the definition of *pullback* and $\mathcal{I}(\Xi, ()G_1, G_2)$. □

Proposition A.162 *If G_1 pullback $G_2 \leq G'_1$ pullback G'_2 , then*

$$\mathcal{I}(\Xi, ()G_1, G_2) = \mathcal{I}(\Xi, ()G_1 \text{ pullback } G_2, G_1 \text{ pullback } G_2) \leq \mathcal{I}(\Xi, ()G'_1 \text{ pullback } G'_2, G'_1 \text{ pullback } G'_2) = \mathcal{I}(\Xi, ()G'_1, G'_2)$$

PROOF. By Proposition A.161 and the definition of \leq in evidence. \square

Proposition A.163 *If $G_1 \leq G_2$, then*

$$\mathcal{I}(\Xi, ()G_1, G_1) \sqsubseteq \mathcal{I}(\Xi, ()G_2, G_2)$$

PROOF. By the definition of $\mathcal{I}(\Xi, ())$ and the \sqsubseteq in evidence. \square

Proposition A.164 $\Omega \vdash \Xi_1 \triangleright s_1 : G_1 \leq \Xi_2 \triangleright s_2 : G_2$ then $G_1 \sqsubseteq G_2$.

PROOF. By the definition of *pullback* and $\mathcal{I}(\Xi, ()G_1, G_2)$. \square

Proposition A.165 *If $\Xi_1 \vdash t_1^* \leq \Xi_2 \vdash t_2^*$ and $\Xi_1 \triangleright t_1^* \longrightarrow \Xi'_1 \triangleright t_1^{**}$, then $\Xi_2 \triangleright t_2^* \longrightarrow \Xi'_2 \triangleright t_2^{**}$ and $\Xi'_1 \vdash t_1^{**} \leq \Xi'_2 \vdash t_2^{**}$.*

PROOF. If $\Xi_1 \vdash t_1^* \leq \Xi_2 \vdash t_2^*$, we know that $\vdash \Xi_1 \triangleright t_1^* : G_1^* \leq \Xi_2 \triangleright t_2^* : G_2^*$, $\Xi_1 \leq \Xi_2$, $\Xi_1 \vdash t_1^* : G_1^*$ and $\Xi_2 \vdash t_2^* : G_2^*$. We follow by induction on $\vdash \Xi_1 \triangleright t_1^* : G_1^* \leq \Xi_2 \triangleright t_2^* : G_2^*$. We avoid the notation $\vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$, and use $t_1 \leq t_2$ instead, for simplicity, when the typing environments are not relevant. We only take into account the existential unpack case.

Case (pack). We know that

$$(\leq_{\text{pack}}) \frac{G_1'' \leq G_2'' \quad \vdash \Xi_1 \triangleright t_{11} : G_{11}[G_1''/X] \leq \Xi_2 \triangleright t_{22} : G_{22}[G_2''/X] \quad \exists X.G_{11} \leq \exists X.G_{22}}{\vdash \Xi_1 \triangleright \text{pack}\langle G_1'', t_{11} \rangle \text{ as } \exists X.G_{11} : \exists X.G_{11} \leq \Xi_2 \triangleright \text{pack}\langle G_2'', t_{22} \rangle \text{ as } \exists X.G_{22} : \exists X.G_{22}}$$

Also, since $\Xi_1 \triangleright t_1^* \longrightarrow \Xi'_1 \triangleright t_1^{**}$, we know that $t_{11} = v_{11}$. By Proposition A.175 and $\vdash \Xi_1 \triangleright t_{11} : G_{11}[G_1''/X] \leq \Xi_2 \triangleright t_{22} : G_{22}[G_2''/X]$, we know that $t_{22} = v_{22}$.

By the reduction rules, we know that

$$\Xi_1 \triangleright \text{pack}\langle G_1'', v_{11} \rangle \text{ as } \exists X.G_{11} \longrightarrow \Xi_1 \triangleright \varepsilon_{\exists X.G_{11}} \text{packu}\langle G_1'', v_{11} \rangle \text{ as } \exists X.G_{11} :: \exists X.G_{11}$$

$$\Xi_2 \triangleright \text{pack}\langle G_2'', v_{22} \rangle \text{ as } \exists X.G_{22} \longrightarrow \Xi_2 \triangleright \varepsilon_{\exists X.G_{22}} \text{packu}\langle G_2'', v_{22} \rangle \text{ as } \exists X.G_{22} :: \exists X.G_{22}$$

We are required to prove that

$$\begin{aligned} & \vdash \Xi_1 \triangleright \varepsilon_{\exists X.G_{11}} \text{packu}\langle G_1'', v_{11} \rangle \text{ as } \exists X.G_{11} :: \exists X.G_{11} \leq \Xi_2 \triangleright : \\ & \varepsilon_{\exists X.G_{22}} \text{packu}\langle G_2'', v_{22} \rangle \text{ as } \exists X.G_{22} :: \exists X.G_{22} : \exists X.G_{11} \leq \exists X.G_{22} \end{aligned}$$

This follows immediately by rules $(\leq_{\text{packu}_\varepsilon})$ and $(\leq_{\text{asc}_\varepsilon})$. Note that $\varepsilon_{\exists X.G_{11}} \leq \varepsilon_{\exists X.G_{22}}$, by Lemma A.163.

Case (unpack). We know that

$$(\leq\text{unpack}) \frac{\vdash \Xi_1 \triangleright t_{11} : \exists X.G_1 \leq \Xi_2 \triangleright t_{21} : \exists X.G_2 \quad x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright t_{12} : G'_1 \leq \Xi_2 \triangleright t_{22} : G'_2}{\vdash \Xi_1 \triangleright \text{unpack}\langle X, x \rangle = t_{11} \text{ in } t_{12} : G'_1 \leq \Xi_2 \triangleright \text{unpack}\langle X, x \rangle = t_{21} \text{ in } t_{22} : G'_2}$$

Also, since $\Xi_1 \triangleright t_1^* \rightarrow \Xi'_1 \triangleright t_1^*$, we know that $t_{11} = \varepsilon_{11} \text{packu}\langle G''_1, \varepsilon_{11} u_1 :: G_{11}[G''_1/X] \rangle$ as $\exists X.G_{11} :: \exists X.G_1$. By Proposition A.175 and $\vdash \Xi_1 \triangleright t_{11} : \exists X.G_1 \leq \Xi_2 \triangleright t_{21} : \exists X.G_2$, we know that $t_{21} = \varepsilon_{22} \text{packu}\langle G''_2, \varepsilon_{22} u_2 :: G_{22}[G''_2/X] \rangle$ as $\exists X.G_{22} :: \exists X.G_2$. By the reduction rules, we know that

$$\Xi_1 \triangleright \text{unpack}\langle X, x \rangle = t_{11} \text{ in } t_{12} \rightarrow \Xi'_1 \triangleright t_{12}[\hat{\alpha}/X][((\varepsilon_1 \circ \varepsilon_{11}[\hat{G}'_1, \hat{\alpha}])u_1 :: G_1[\alpha/X])/x]$$

where $\Xi'_1 = \Xi_1$, $\alpha := G''_1$ and $\hat{\alpha}_1 = \text{lift}_{\Xi'_1}(\alpha)$.

We know that $\varepsilon_{11} \leq \varepsilon_{22}$, $\Xi'_1 \leq \Xi'_2$ and $G''_1 \leq G''_2$, therefore by Proposition A.167, we know that $\varepsilon_{11}[\hat{G}'_1, \hat{\alpha}] \leq \varepsilon_{22}[\hat{G}'_2, \hat{\alpha}]$. Therefore, we know that $(\varepsilon_1 \circ \varepsilon_{11}[\hat{G}'_1, \hat{\alpha}]) \leq (\varepsilon_2 \circ \varepsilon_{22}[\hat{G}'_2, \hat{\alpha}])$, by Proposition A.168 and $\varepsilon_1 \leq \varepsilon_2$.

Therefore, we know that

$$\Xi_2 \triangleright \text{unpack}\langle X, x \rangle = t_{21} \text{ in } t_{22} \rightarrow \Xi'_2 \triangleright t_{22}[\hat{\alpha}/X][((\varepsilon_2 \circ \varepsilon_{22}[\hat{G}'_2, \hat{\alpha}])u_2 :: G_2[\alpha/X])/x]$$

where $\Xi'_2 = \Xi_2$, $\alpha := G''_2$ and $\hat{\alpha}_2 = \text{lift}_{\Xi'_2}(\alpha)$.

Since $\Xi_1 \leq \Xi_2$ and $G''_1 \leq G''_2$, we know that $\Xi'_1 \leq \Xi'_2$. Therefore, we only are required to prove that

$$t_{12}[\hat{\alpha}/X][((\varepsilon_1 \circ \varepsilon_{11}[\hat{G}'_1, \hat{\alpha}])u_1 :: G_1[\alpha/X])/x] : G'_1 \leq t_{22}[\hat{\alpha}/X][((\varepsilon_2 \circ \varepsilon_{22}[\hat{G}'_2, \hat{\alpha}])u_2 :: G_2[\alpha/X])/x] : G'_2$$

By Proposition A.169 we know that $t_{12}[\hat{\alpha}_1/X] \leq t_{22}[\hat{\alpha}_2/X]$.

We know that $((\varepsilon_1 \circ \varepsilon_{11}[\hat{G}'_1, \hat{\alpha}])u_1 :: G_1[\alpha/X]) \leq ((\varepsilon_2 \circ \varepsilon_{22}[\hat{G}'_2, \hat{\alpha}])u_2 :: G_2[\alpha/X])$, by the Rule $(\leq\text{asc}_\varepsilon)$ and since $u_1 \leq u_2$, $(\varepsilon_1 \circ \varepsilon_{11}[\hat{G}'_1, \hat{\alpha}]) \leq (\varepsilon_2 \circ \varepsilon_{22}[\hat{G}'_2, \hat{\alpha}])$ and $G_1[\alpha/X] \sqsubseteq G_2[\alpha/X]$ (by Proposition A.172 and Proposition A.173). Finally, by Proposition A.166 the result holds. \square

Proposition A.166 (Substitution Preserves Precision) *If $\Omega', x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright s_1 : G'_1 \leq \Xi_2 \triangleright s_2 : G'_2$ and $\Omega' \vdash \Xi_1 \triangleright v_1 : G_1 \leq \Xi_2 \triangleright v_2 : G_2$, then $\Omega' \vdash \Xi_1 \triangleright s_1[v_1/x] : G'_1 \leq \Xi_2 \triangleright s_2[v_2/x] : G'_2$.*

PROOF. We follow by induction on $\Omega', x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright t_1 : G'_1 \leq \Xi_2 \triangleright t_2 : G'_2$. We avoid the notation $\Omega', x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright t_1 : G'_1 \leq \Xi_2 \triangleright t_2 : G'_2$, and use $t_1 \leq t_2$ instead, for simplicity, when the typing environments are not relevant. Let suppose that $\Omega = \Omega', x : G_1 \sqsubseteq G_2$.

Case (packu). We know that

$$(\leq\text{packu}_\varepsilon) \frac{G_1^{**} \leq G_2^{**} \quad \Omega \vdash \Xi_1 \triangleright v'_1 : G_1^*[G_1^{**}/X] \leq \Xi_2 \triangleright v'_2 : G_2^*[G_2^{**}/X] \quad \exists X.G_1^* \sqsubseteq \exists X.G_2^*}{\Omega \vdash \Xi_1 \triangleright \text{packu}\langle G_1^{**}, v'_1 \rangle \text{ as } \exists X.G_1^* : \exists X.G_1^* \leq \Xi_2 \triangleright \text{packu}\langle G_2^{**}, v'_2 \rangle \text{ as } \exists X.G_2^* : \exists X.G_2^*}$$

Note that we are required to prove that

$$\Omega \vdash \Xi_1 \triangleright \text{packu}\langle G_1^{**}, v'_1[v_1/x] \rangle \text{ as } \exists X.G_1^* : \exists X.G_1^* \leq \Xi_2 \triangleright \text{packu}\langle G_2^{**}, v'_2[v_2/x] \rangle \text{ as } \exists X.G_2^* : \exists X.G_2^*$$

or what is the same $\Omega \vdash \Xi_1 \triangleright v'_1[v_1/x] : G_1^*[G_1^{**}/X] \leq \Xi_2 \triangleright v'_2[v_2/x] : G_2^*[G_2^{**}/X]$. But the result follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright v'_1 : G_1^*[G_1^{**}/X] \leq \Xi_2 \triangleright v'_2 : G_2^*[G_2^{**}/X]$.

Case (pack). We know that

$$(\leq_{\text{pack}_\varepsilon}) \frac{G_1^{**} \leq G_2^{**} \quad \Omega \vdash \Xi_1 \triangleright t_1 : G_1^*[G_1^{**}/X] \leq \Xi_2 \triangleright t_2 : G_2^*[G_2^{**}/X] \quad \exists X.G_1^* \leq \exists X.G_2^*}{\Omega \vdash \Xi_1 \triangleright \text{pack}\langle G_1^{**}, t_1 \rangle \text{ as } \exists X.G_1^* : \exists X.G_1^* \leq \Xi_2 \triangleright \text{pack}\langle G_2^{**}, t_2 \rangle \text{ as } \exists X.G_2^* : \exists X.G_2^*}$$

Note that we are required to prove that

$$\Omega \vdash \Xi_1 \triangleright \text{pack}\langle G_1^{**}, t_1[v_1/x] \rangle \text{ as } \exists X.G_1^* : \exists X.G_1^* \leq \Xi_2 \triangleright \text{pack}\langle G_2^{**}, t_2[v_2/x] \rangle \text{ as } \exists X.G_2^* : \exists X.G_2^*$$

or what is the same $\Omega \vdash \Xi_1 \triangleright t_1[v_1/x] : G_1^*[G_1^{**}/X] \leq \Xi_2 \triangleright t_2[v_2/x] : G_2^*[G_2^{**}/X]$. But the result follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t_1 : G_1^*[G_1^{**}/X] \leq \Xi_2 \triangleright t_2 : G_2^*[G_2^{**}/X]$.

Case (unpack). We know that

$$(\leq_{\text{unpack}_\varepsilon}) \frac{\Omega \vdash \Xi_1 \triangleright t_1 : \exists X.G_1^* \leq \Xi_2 \triangleright t_2 : \exists X.G_2^* \quad \Omega, x : G_1^* \sqsubseteq G_2^* \vdash \Xi_1 \triangleright t'_1 : G_1^{**} \leq \Xi_2 \triangleright t'_2 : G_2^{**}}{\Omega \vdash \Xi_1 \triangleright \text{unpack}\langle X, x \rangle = t_1 \text{ in } t'_1 : G_1^{**} \leq \Xi_2 \triangleright \text{unpack}\langle X, x \rangle = t_2 \text{ in } t'_2 : G_2^{**}}$$

Note that we are required to prove that $\Omega' \vdash \Xi_1 \triangleright \text{unpack}\langle X, x \rangle = t_1[v_1/x]$ in $t'_1[v_1/x] : G_1^{**} \leq \Xi_2 \triangleright \text{unpack}\langle X, x \rangle = t_2[v_2/x]$ in $t'_2[v_2/x] : G_2^{**}$. Or what is the same $\Omega' \vdash \Xi_1 \triangleright t_1[v_1/x] : \exists X.G_1^* \leq \Xi_2 \triangleright t_2[v_2/x] : \exists X.G_2^*$ and $\Omega', x : G_1^* \sqsubseteq G_2^* \vdash \Xi_1 \triangleright t'_1[v_1/x] : G_1^{**} \leq \Xi_2 \triangleright t'_2[v_2/x] : G_2^{**}$. But the result follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t_1 : \exists X.G_1^* \leq \Xi_2 \triangleright t_2 : \exists X.G_2^*$ and $\Omega, x : G_1^* \sqsubseteq G_2^* \vdash \Xi_1 \triangleright t'_1 : G_1^{**} \leq \Xi_2 \triangleright t'_2 : G_2^{**}$.

□

Proposition A.167 *If $\varepsilon_1 \leq \varepsilon_2$, $G_1 \leq G_2$, $\Xi_1 \leq \Xi_2$, $\alpha := G_1 \in \Xi_1$, $\alpha := G_2 \in \Xi_2$ and $\varepsilon_1[\hat{G}_1, \hat{\alpha}_1]$ is defined, then $\varepsilon_1[\hat{G}_1, \hat{\alpha}_1] \leq \varepsilon_2[\hat{G}_2, \hat{\alpha}_2]$, where $\hat{\alpha}_1 = \text{lift}_{\Xi_1}(\alpha)$, $\hat{\alpha}_2 = \text{lift}_{\Xi_2}(\alpha)$, $\hat{G}_1 = \text{lift}_{\Xi_1}(G_1)$ and $\hat{G}_2 = \text{lift}_{\Xi_2}(G_2)$.*

PROOF. Note that $\hat{\alpha}_1 \leq \hat{\alpha}_2$ and $\hat{G}_1 \leq \hat{G}_2$ by Proposition A.170. Suppose that $\varepsilon_1 = \langle \exists X.E, \exists X.E' \rangle$ and $\varepsilon_2 = \langle \exists X.E'', \exists X.E''' \rangle$ (since $\varepsilon_1[\hat{G}_1, \hat{\alpha}_1]$ is defined). We are required to prove that

$$\varepsilon_1[\hat{G}_1, \hat{\alpha}_1] = \langle E[\hat{G}_1/X], E'[\hat{\alpha}_1/X] \rangle \leq \langle E''[\hat{G}_2/X], E'''[\hat{\alpha}_2/X] \rangle = \varepsilon_2[\hat{G}_2, \hat{\alpha}_2]$$

Thus, we are required to prove that $E[\hat{G}_1/X] \leq E''[\hat{G}_2/X]$ and $E'[\hat{\alpha}_1/X] \leq E'''[\hat{\alpha}_2/X]$. Since $\varepsilon_1 \leq \varepsilon_2$, we know that $\langle \exists X.E, \exists X.E' \rangle \leq \langle \exists X.E'', \exists X.E''' \rangle$, and therefore $E \leq E''$ and $E' \leq E'''$. By Proposition A.174 and $\hat{\alpha}_1 \leq \hat{\alpha}_2$ and $\hat{G}_1 \leq \hat{G}_2$, we know that $E[\hat{G}_1/X] \leq E''[\hat{G}_2/X]$ and $E'[\hat{\alpha}_1/X] \leq E'''[\hat{\alpha}_2/X]$. Therefore the result holds. □

Proposition A.168 (Monotonicity of Evidence Transitivity) *If $\varepsilon_1 \leq \varepsilon_2$, $\varepsilon_3 \leq \varepsilon_4$, and $\varepsilon_1 \circ \varepsilon_3$ is defined, then $\varepsilon_1 \circ \varepsilon_3 \leq \varepsilon_2 \circ \varepsilon_4$.*

PROOF. By definition of consistent transitivity for $=$ and the definition of precision. We only take into account the existential type case.

Case ($[\exists]$ - $\varepsilon_i = \langle \exists X.E_i, \exists X.E'_i \rangle$). By the definition of \leq , we know that $\langle E_1, E'_1 \rangle \leq \langle E_2, E'_2 \rangle$ and $\langle E_3, E'_3 \rangle \leq \langle E_4, E'_4 \rangle$. By the definition of transitivity we know that $\langle \exists X.E_1, \exists X.E'_1 \rangle \wp \langle \exists X.E_3, \exists X.E'_3 \rangle = \langle \exists X.E_5, \exists X.E'_5 \rangle$ and $\langle \exists X.E_2, \exists X.E'_2 \rangle \wp \langle \exists X.E_4, \exists X.E'_4 \rangle = \langle \exists X.E_6, \exists X.E'_6 \rangle$, where $\langle E_5, E'_5 \rangle = \langle E_1, E'_1 \rangle \wp \langle E_3, E'_3 \rangle$ and $\langle E_6, E'_6 \rangle = \langle E_2, E'_2 \rangle \wp \langle E_4, E'_4 \rangle$. Therefore, we are required to prove that $\langle E_5, E'_5 \rangle \leq \langle E_6, E'_6 \rangle$. But the result follows immediately by the induction hypothesis on $\langle E_1, E'_1 \rangle \leq \langle E_2, E'_2 \rangle$ and $\langle E_3, E'_3 \rangle \leq \langle E_4, E'_4 \rangle$.

□

Proposition A.169 (Monotonicity of Evidence Substitution) *If $\Omega \vdash \Xi_1 \triangleright s_1^* : G_1^* \leq \Xi_2 \triangleright s_2^* : G_2^*$ and $\Xi_1 \leq \Xi_2$, then $\Omega[\alpha/X] \vdash \Xi_1 \triangleright s_1^*[\hat{\alpha}_1/X] : G_1^*[\alpha/X] \leq \Xi_2 \triangleright s_2^*[\hat{\alpha}_2/X] : G_2^*[\alpha/X]$, where $\alpha := G_1^{**} \in \Xi_1$, $\alpha := G_2^{**} \in \Xi_2$, $\hat{\alpha}_1 = \text{lift}_{\Xi_1}(\alpha)$ and $\hat{\alpha}_2 = \text{lift}_{\Xi_2}(\alpha)$.*

PROOF. We follow by induction on $\Omega \vdash \Xi_1 \triangleright s_1^* : G_1^* \leq \Xi_2 \triangleright s_2^* : G_2^*$. We avoid the notation $\Omega \vdash \Xi_1 \triangleright s_1^* : G_1^*[\alpha/X] \leq \Xi_2 \triangleright s_2^* : G_2^*[\alpha/X]$, and use $s_1^* \leq s_2^*$ instead, for simplicity, when the typing environments are not relevant. We only take into account the cases related to existential types.

Case (packu). We know that

$$(\leq_{\text{packu}_\varepsilon}) \frac{G'_1 \leq G'_2 \quad \Omega \vdash \Xi_1 \triangleright v_1 : G_1[G'_1/Y] \leq \Xi_2 \triangleright v_2 : G_2[G'_2/Y] \quad \exists Y.G_1 \sqsubseteq \exists Y.G_2}{\Omega \vdash \Xi_1 \triangleright \text{packu}\langle G'_1, v_1 \rangle \text{ as } \exists Y.G_1 : \exists Y.G_1 \leq \Xi_2 \triangleright \text{packu}\langle G'_2, v_2 \rangle \text{ as } \exists Y.G_2 : \exists Y.G_2}$$

We are required to show

$$\begin{aligned} \Omega[\alpha/X] \vdash \Xi_1 \triangleright \text{packu}\langle G'_1[\alpha/X], v_1[\hat{\alpha}_1/X] \rangle \text{ as } \exists Y.G_1[\alpha/X] &: \leq \Xi_2 \triangleright : \\ \text{packu}\langle G'_2, v_2[\hat{\alpha}_1/X] \rangle \text{ as } \exists Y.G_2 : \exists Y.G_1[\alpha/X] &\leq \exists Y.G_2[\alpha/X] \end{aligned}$$

Note that $G'_1[\alpha/X] \leq G'_2[\alpha/X]$ by Proposition A.174 and $\exists Y.G_1[\alpha/X] \sqsubseteq \exists Y.G_2[\alpha/X]$ by Proposition A.173. Therefore, we are required to prove $\Omega[\alpha/X] \vdash \Xi_1 \triangleright (v_1[\hat{\alpha}_1/X]) : G_1[G'_1/Y][\alpha/X] \leq \Xi_2 \triangleright (v_2[\hat{\alpha}_2/X]) : G_2[G'_2/Y][\alpha/X]$. But the result follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright v_1 : G_1[G'_1/Y] \leq \Xi_2 \triangleright v_2 : G_2[G'_2/Y]$.

Case (pack). We know that

$$(\leq_{\text{pack}_\varepsilon}) \frac{G'_1 \leq G'_2 \quad \Omega \vdash \Xi_1 \triangleright t_1 : G_1[G'_1/Y] \leq \Xi_2 \triangleright t_2 : G_2[G'_2/Y] \quad \exists Y.G_1 \leq \exists Y.G_2}{\Omega \vdash \Xi_1 \triangleright \text{pack}\langle G'_1, t_1 \rangle \text{ as } \exists Y.G_1 : \exists Y.G_1 \leq \Xi_2 \triangleright \text{pack}\langle G'_2, t_2 \rangle \text{ as } \exists Y.G_2 : \exists Y.G_2}$$

We are required to show

$$\begin{aligned} \Omega[\alpha/X] \vdash \Xi_1 \triangleright \text{pack}\langle G'_1[\alpha/X], t_1[\hat{\alpha}_1/X] \rangle \text{ as } \exists Y.G_1[\alpha/X] &: \leq \Xi_2 \triangleright : \\ \text{pack}\langle G'_2, t_2[\hat{\alpha}_1/X] \rangle \text{ as } \exists Y.G_2 : \exists Y.G_1[\alpha/X] &\leq \exists Y.G_2[\alpha/X] \end{aligned}$$

Note that $G'_1[\alpha/X] \leq G'_2[\alpha/X]$ by Proposition A.174 and $\exists Y.G_1[\alpha/X] \leq \exists Y.G_2[\alpha/X]$ by Proposition A.174. Therefore, we are required to prove $\Omega[\alpha/X] \vdash \Xi_1 \triangleright (t_1[\hat{\alpha}_1/X]) : G_1[G'_1/Y][\alpha/X] \leq \Xi_2 \triangleright (t_2[\hat{\alpha}_2/X]) : G_2[G'_2/Y][\alpha/X]$. But the result follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t_1 : G_1[G'_1/Y] \leq \Xi_2 \triangleright t_2 : G_2[G'_2/Y]$.

Case (unpack). We know that

$$(\leq_{\text{unpack}_\varepsilon}) \frac{\Omega \vdash \Xi_1 \triangleright t_1 : \exists Y.G_1 \leq \Xi_2 \triangleright t_2 : \exists Y.G_2 \quad \Omega, x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright t'_1 : G'_1 \leq \Xi_2 \triangleright t'_2 : G'_2}{\Omega \vdash \Xi_1 \triangleright \text{unpack}\langle Y, x \rangle = t_1 \text{ in } t'_1 : G'_1 \leq \Xi_2 \triangleright \text{unpack}\langle Y, x \rangle = t_2 \text{ in } t'_2 : G'_2}$$

We are required to show

$$\Omega[\alpha/X] \vdash \Xi_1 \triangleright \text{unpack}\langle Y, x \rangle = t_1[\hat{\alpha}_1/X] \text{ in } t'_1[\hat{\alpha}_1/X] : G'_1[\alpha/X] \leq \Xi_2 \triangleright \text{unpack}\langle Y, x \rangle = t_2[\hat{\alpha}_2/X] \text{ in } t'_2[\hat{\alpha}_2/X] : G'_2[\alpha/X]$$

Therefore, we are required to prove $\Omega[\alpha/X] \vdash \Xi_1 \triangleright (t_1[\hat{\alpha}_1/X]) : \exists Y.G_1[\alpha/X] \leq \Xi_2 \triangleright (t_2[\hat{\alpha}_2/X]) : \exists Y.G_2[\alpha/X]$ and $\Omega[\alpha/X], x : G_1[\alpha/X] \sqsubseteq G_2[\alpha/X] \vdash \Xi_1 \triangleright (t'_1[\hat{\alpha}_1/X]) : G'_1[\alpha/X] \leq \Xi_2 \triangleright (t'_2[\hat{\alpha}_2/X]) : G'_2[\alpha/X]$. But the results follows immediately by the induction hypothesis on $\Omega \vdash \Xi_1 \triangleright t_1 : \exists Y.G_1 \leq \Xi_2 \triangleright t_2 : \exists Y.G_2$ and $\Omega, x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright t'_1 : G'_1 \leq \Xi_2 \triangleright t'_2 : G'_2$.

□

Proposition A.170 (Lift Environment Precision) *If $G_1 \leq G_2$ and $\Xi_1 \leq \Xi_2$, then $\hat{G}_1 \leq \hat{G}_2$, where $\hat{G}_1 = \text{lift}_{\Xi_1}(G_1)$ and $\hat{G}_2 = \text{lift}_{\Xi_2}(G_2)$.*

PROOF. Remember that

$$\text{lift}_{\Xi}(G) = \begin{cases} \text{lift}_{\Xi}(G_1) \rightarrow \text{lift}_{\Xi}(G_2) & G = G_1 \rightarrow G_2 \\ \forall X. \text{lift}_{\Xi}(G_1) & G = \forall X.G_1 \\ \exists X. \text{lift}_{\Xi}(G_1) & G = \exists X.G_1 \\ \text{lift}_{\Xi}(G_1) \times \text{lift}_{\Xi}(G_2) & G = G_1 \times G_2 \\ \alpha^{\text{lift}_{\Xi}(\Xi(\alpha))} & G = \alpha \\ G & \text{otherwise} \end{cases}$$

The prove follows by the definition of $\hat{G}_1 = \text{lift}_{\Xi_1}(G_1)$ and induction on the structure of the type.

Case ($G_i = \exists X.G'_i$). We know that $G'_1 \leq G'_2$. We are required to prove that $\exists X. \text{lift}_{\Xi_1}(G'_1) \leq \exists X. \text{lift}_{\Xi_2}(G'_2)$, or what is the same $\text{lift}_{\Xi_1}(G'_1) \leq \text{lift}_{\Xi_2}(G'_2)$. By the induction hypothesis on $G'_1 \leq G'_2$ and $\Xi_1 \leq \Xi_2$ the result follows immediately.

□

Proposition A.171 *If $G_1^* \leq G_2^*$ then $G_1^* \sqsubseteq G_2^*$.*

PROOF. Examining \leq rules.

Case ($\exists X.G_1 \leq \exists X.G_2$). We know that

$$\frac{G_1 \leq G_2}{\exists X.G_1 \leq \exists X.G_2}$$

By the induction hypothesis on $G_1 \leq G_2$, we know that $G_1 \sqsubseteq G_2$. We are required to prove that $\exists X.G_1 \sqsubseteq \exists X.G_2$, which follows immediately by the rule

$$\frac{G_1 \sqsubseteq G_2}{\exists X.G_1 \sqsubseteq \exists X.G_2}$$

□

Proposition A.172 *If $G_1^* \sqsubseteq G_2^*$ and $G_1' \sqsubseteq G_2'$ then $G_1^*[G_1'/X] \sqsubseteq G_2^*[G_2'/X]$.*

PROOF. Follow by induction on $G_1^* \sqsubseteq G_2^*$. We only take into account the existential type case.

Case $(\exists X.G_1 \sqsubseteq \exists X.G_2)$. We know that

$$\frac{G_1 \sqsubseteq G_2}{\exists X.G_1 \sqsubseteq \exists X.G_2}$$

By the definition of \sqsubseteq , we know that $G_1 \sqsubseteq G_2$. We are required to prove that

$$(\exists X.G_1)[G_1'/X] = (\exists X.G_1[G_1'/X]) \sqsubseteq (\exists X.G_2[G_2'/X]) = (\exists X.G_2)[G_2'/X]$$

Or what is the same that $(G_1[G_1'/X]) \sqsubseteq (G_2[G_2'/X])$. But the result follows immediately by the induction hypothesis on $G_1 \sqsubseteq G_2$.

□

Proposition A.173 *If $G_1 \sqsubseteq G_2$ and $G_1' \leq G_2'$ then $G_1[G_1'/X] \sqsubseteq G_2[G_2'/X]$.*

PROOF. By Proposition A.171 and Proposition A.172 the results follows immediately.

□

Proposition A.174 *If $G_1 \leq G_2$ and $G_1' \leq G_2'$ then $G_1[G_1'/X] \leq G_2[G_2'/X]$.*

PROOF. Straightforward induction on $G_1 \leq G_2$. Very similar to Proposition A.172.

□

Proposition A.175 *If $v_1 \leq t_2$ then $t_2 = v_2$.*

PROOF. Exploring \leq rules.

□

Proposition A.176 *If $\Xi_1 \vdash t_1 \leq \Xi_2 \vdash t_2$ and $\Xi_1 \triangleright t_1 \mapsto \Xi_1' \triangleright t_1'$, then $\Xi_2 \triangleright t_2 \mapsto \Xi_2' \triangleright t_2'$ and $\Xi_1' \vdash t_1' \leq \Xi_2' \vdash t_2'$.*

PROOF. If $\Xi_1 \vdash t_1 \leq \Xi_2 \vdash t_2$, we know that $\vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$, $\Xi_1 \leq \Xi_2$, $\Xi_1 \vdash t_1 : G_1$ and $\Xi_2 \vdash t_2 : G_2$. We avoid the notation $\vdash \Xi_1 \triangleright t_1 : G_1 \leq \Xi_2 \triangleright t_2 : G_2$, and use $t_1 \leq t_2$ instead, for simplicity, when the typing environments are not relevant.

By induction on reduction $\Xi_1 \triangleright t_1 \mapsto \Xi_1' \triangleright t_1'$. We only take into account the existential unpack case.

Case $(\Xi_1 \triangleright \text{unpack}\langle X, x \rangle = t_{11} \text{ in } t_{12} \mapsto \Xi_1' \triangleright \text{unpack}\langle X, x \rangle = t_{11}' \text{ in } t_{12})$. By inspection of \leq , $t_2 = \text{unpack}\langle X, x \rangle = t_{21} \text{ in } t_{22}$, where $t_{11} \leq t_{21}$ and $t_{12} \leq t_{22}$. By induction hypothesis on $\Xi_1 \triangleright t_{11} \mapsto \Xi_1' \triangleright t_{11}'$, we know that $\Xi_2 \triangleright t_{21} \mapsto \Xi_2' \triangleright t_{21}'$, where $\Xi_1' \vdash t_{11}' \leq \Xi_2' \vdash t_{21}'$. Then, by \leq , we know that $\Xi_1' \triangleright \text{unpack}\langle X, x \rangle = t_{11}' \text{ in } t_{12} \leq \Xi_2' \triangleright \text{unpack}\langle X, x \rangle = t_{21}' \text{ in } t_{22}$ and the result holds.

□

$G \leq G$ **Strict type precision**

$$\frac{G_1 \leq G_2}{\exists X.G_1 \leq \exists X.G_2}$$

$\Omega \vdash \Xi_1 \triangleright s : G \leq \Xi_2 \triangleright s : G$ **Strict term precision** (for conciseness, s ranges over both t and u)

$$\begin{aligned} (\leq \text{packu}_\epsilon) \frac{G'_1 \leq G'_2 \quad \Omega \vdash \Xi_1 \triangleright v_1 : G_1[G'_1/X] \leq \Xi_2 \triangleright v_2 : G_2[G'_2/X] \quad \exists X.G_1 \sqsubseteq \exists X.G_2}{\Omega \vdash \Xi_1 \triangleright \text{packu}\langle G'_1, v_1 \rangle \text{ as } \exists X.G_1 : \exists X.G_1 \leq \Xi_2 \triangleright \text{packu}\langle G'_2, v_2 \rangle \text{ as } \exists X.G_2 : \exists X.G_2} \\ (\leq \text{pack}_\epsilon) \frac{G'_1 \leq G'_2 \quad \Omega \vdash \Xi_1 \triangleright t_1 : G_1[G'_1/X] \leq \Xi_2 \triangleright t_2 : G_2[G'_2/X] \quad \exists X.G_1 \leq \exists X.G_2}{\Omega \vdash \Xi_1 \triangleright \text{pack}\langle G'_1, t_1 \rangle \text{ as } \exists X.G_1 : \exists X.G_1 \leq \Xi_2 \triangleright \text{pack}\langle G'_2, t_2 \rangle \text{ as } \exists X.G_2 : \exists X.G_2} \\ (\leq \text{unpack}_\epsilon) \frac{\Omega \vdash \Xi_1 \triangleright t_1 : \exists X.G_1 \leq \Xi_2 \triangleright t_2 : \exists X.G_2 \quad \Omega, x : G_1 \sqsubseteq G_2 \vdash \Xi_1 \triangleright t'_1 : G'_1 \leq \Xi_2 \triangleright t'_2 : G'_2}{\Omega \vdash \Xi_1 \triangleright \text{unpack}\langle X, x \rangle = t_1 \text{ in } t'_1 : G'_1 \leq \Xi_2 \triangleright \text{unpack}\langle X, x \rangle = t_2 \text{ in } t'_2 : G'_2} \end{aligned}$$

$G \rightarrow G$ **Type matching**

$$? \rightarrow \exists X.?$$

$\Omega \vdash v : G \leq_v v : G$ **Strict value precision**

$$(\leq \text{packu}) \frac{G'_1 \leq G'_2 \quad \Omega \vdash v_1 : G''_1 \leq_v v_2 : G''_2 \quad \exists X.G_1 \sqsubseteq \exists X.G_2 \quad G''_1 \text{pullback}G_1[G'_1/X] \leq G''_2 \text{pullback}G_2[G'_2/X]}{\Omega \vdash \text{pack}\langle G'_1, v_1 \rangle \text{ as } \exists X.G_1 : \exists X.G_1 \leq_v \text{pack}\langle G'_2, v_2 \rangle \text{ as } \exists X.G_2 : \exists X.G_2}$$

$\Omega \vdash t : G \leq t : G$ **Strict term precision**

$$\begin{aligned} (\leq \text{pack}) \frac{G'_1 \leq G'_2 \quad \Omega \vdash t_1 : G''_1 \leq t_2 : G''_2 \quad \exists X.G_1 \leq \exists X.G_2 \quad G''_1 \text{pullback}G_1[G'_1/X] \leq G''_2 \text{pullback}G_2[G'_2/X]}{\Omega \vdash \text{pack}\langle G'_1, t_1 \rangle \text{ as } \exists X.G_1 : \exists X.G_1 \leq \text{pack}\langle G'_2, t_2 \rangle \text{ as } \exists X.G_2 : \exists X.G_2} \\ (\leq \text{unpack}) \frac{\Omega \vdash t_1 : G_1 \leq t_2 : G_2 \quad \Omega, x : \text{schm}_e^\sharp(G_1) \sqsubseteq \text{schm}_e^\sharp(G_2) \vdash t'_1 : G'_1 \leq t'_2 : G'_2}{\Omega \vdash \text{unpack}\langle X, x \rangle = t_1 \text{ in } t'_1 : G'_1 \leq \text{unpack}\langle X, x \rangle = t_2 \text{ in } t'_2 : G'_2} \end{aligned}$$

Figure A.8: $\text{GSF}_\epsilon^\exists$ and GSF^\exists : Strict term precision

Annex B

Plausible sealing

This chapter describes all the definitions and demonstrations omitted in the Plausible Sealing part.

B.1 The Evidence-Based Language $F_\varepsilon^?$

This section presents the complete definition of $F_\varepsilon^?$, the type safety proof, and some definitions such as type substitution and well-formedness.

$$\begin{aligned} ?_\delta[G'/X] &= ?_\delta[G'/X] \\ B[G'/X] &= B \\ X[G'/X] &= G' \\ Y[G'/X] &= Y \\ G_1 \rightarrow G_2[G'/X] &= G_1[G'/X] \rightarrow G_2[G'/X] \\ G_1 \times G_2[G'/X] &= G_1[G'/X] \times G_2[G'/X] \\ \forall Y. G[G'/X] &= \forall Y. G[G'/X] \end{aligned}$$

Substitution of a type variable in δ works as follows:

$$\begin{aligned} \cdot[G/X] &= \cdot \\ (\delta, Y : F)[G/X] &= \delta[G/X], Y : F[G/X] \end{aligned}$$

Definition B.1 (Well-formedness of types)

$$\begin{array}{c} \frac{}{\Delta \vdash B} \quad \frac{X \in \Delta}{\Delta \vdash X} \quad \frac{\Delta \vdash G_1 \quad \Delta \vdash G_2}{\Delta \vdash G_1 \rightarrow G_2} \quad \frac{\Delta \vdash G_1 \quad \Delta \vdash G_2}{\Delta \vdash G_1 \times G_2} \\ \\ \frac{\Delta, X \vdash G}{\Delta \vdash \forall X. G} \quad \frac{\Delta \vdash \delta}{\Delta \vdash ?_\delta} \end{array}$$

$\mathbf{X} \in \text{TYPEVAR}$, $\mathbf{G} \in \text{GTYPE}$, $\varepsilon \in \text{EVIDENCE}$, $\mathbf{t} \in \text{TERM}$, $\Delta \subset \text{TYPEVAR}$, $\Gamma \in \text{VAR} \xrightarrow{\text{fin}} \text{GTYPE}$
 $\mathbf{G} ::= \mathbf{B} \mid \mathbf{X} \mid \mathbf{G} \rightarrow \mathbf{G} \mid \mathbf{G} \times \mathbf{G} \mid \forall \mathbf{X}. \mathbf{G} \mid ?_{\delta}$ (gradual types)
 $\mathbf{F} ::= \mathbf{B} \mid \mathbf{X}$ (instantiation types)
 $\delta ::= \delta, \mathbf{X} : \mathbf{F} \mid \emptyset$ (instantiation environment)
 $\mathbf{t} ::= \mathbf{v} \mid \mathbf{x} \mid \mathbf{t} \mathbf{t} \mid \langle \mathbf{t}, \mathbf{t} \rangle \mid \pi_i(\mathbf{t}) \mid \mathbf{t} [\mathbf{F}] \mid \varepsilon \mathbf{t} :: \mathbf{G}$ (terms)
 $\mathbf{v} ::= \varepsilon \mathbf{u} :: \mathbf{G}$ (values)
 $\mathbf{u} ::= \mathbf{b} \mid \lambda \mathbf{x} : \mathbf{G}. \mathbf{t} \mid \langle \mathbf{u}, \mathbf{u} \rangle \mid \Lambda \mathbf{X}. \mathbf{t}$ (raw values)
 $\mathbf{s} ::= \mathbf{u} \mid \mathbf{t}$

$\Delta; \Gamma \vdash \mathbf{s} : \mathbf{G}$ Term typing

$$\begin{array}{c}
\text{Gb} \frac{\theta(\mathbf{b}) = \mathbf{B} \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \mathbf{b} : \mathbf{B}} \qquad \text{G}_{\lambda} \frac{\Delta; \Gamma, \mathbf{x} : \mathbf{G}_1 \vdash \mathbf{t} : \mathbf{G}_2}{\Delta; \Gamma \vdash \lambda \mathbf{x} : \mathbf{G}_1. \mathbf{t} : \mathbf{G}_1 \rightarrow \mathbf{G}_2} \\
\text{G}_{\times} \frac{\Delta; \Gamma \vdash \mathbf{s}_1 : \mathbf{G}_1 \quad \Delta; \Gamma \vdash \mathbf{s}_2 : \mathbf{G}_2}{\Delta; \Gamma \vdash \langle \mathbf{s}_1, \mathbf{s}_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2} \qquad \text{G}_{\Lambda} \frac{\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t} : \mathbf{G} \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \Lambda \mathbf{X}. \mathbf{t} : \forall \mathbf{X}. \mathbf{G}} \\
\text{G}_{\mathbf{x}} \frac{\mathbf{x} : \mathbf{G} \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \mathbf{x} : \mathbf{G}} \qquad \text{G}_{\text{asc}} \frac{\Delta; \Gamma \vdash \mathbf{s} : \mathbf{G}' \quad \Delta \vdash \mathbf{G} \quad \varepsilon : \mathbf{G}' \sim \mathbf{G}}{\Delta; \Gamma \vdash \varepsilon \mathbf{s} :: \mathbf{G} : \mathbf{G}} \\
\text{G}_{\text{app}} \frac{\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \quad \Delta; \Gamma \vdash \mathbf{t}_2 : \mathbf{G}_1}{\Delta; \Gamma \vdash \mathbf{t}_1 \mathbf{t}_2 : \mathbf{G}_2} \qquad \text{G}_{\text{pair}_i} \frac{\Delta; \Gamma \vdash \mathbf{t} : \mathbf{G}_1 \times \mathbf{G}_2}{\Delta; \Gamma \vdash \pi_i(\mathbf{t}) : \mathbf{G}_i} \\
\text{G}_{\text{appG}} \frac{\Delta; \Gamma \vdash \mathbf{t} : \forall \mathbf{X}. \mathbf{G} \quad \Delta \vdash \mathbf{F}}{\Delta; \Gamma \vdash \mathbf{t} [\mathbf{F}] : \mathbf{G}[\mathbf{F}/\mathbf{X}]}
\end{array}$$

$\mathbf{t} \longrightarrow \mathbf{t}$ or error Notion of reduction

$$\begin{array}{l}
(\text{Rasc}) \quad \varepsilon_2 (\varepsilon_1 \mathbf{u} :: \mathbf{G}_1) :: \mathbf{G}_2 \longrightarrow \begin{cases} \varepsilon \mathbf{u} :: \mathbf{G}_2 & \text{if } \varepsilon = \varepsilon_1 \mathbin{\&}; \varepsilon_2 \\ \mathbf{error} & \text{otherwise} \end{cases} \\
(\text{Rapp}) \quad \begin{array}{l} (\varepsilon_1 (\lambda \mathbf{x} : \mathbf{G}_{11}. \mathbf{t}) :: \mathbf{G}_1 \rightarrow \mathbf{G}_2) \\ (\varepsilon_2 \mathbf{u} :: \mathbf{G}_1) \end{array} \longrightarrow \begin{cases} \text{cod}(\varepsilon_1) (\mathbf{t}[(\varepsilon \mathbf{u} :: \mathbf{G}_{11})/\mathbf{x}]) :: \mathbf{G}_2 & \text{if } \varepsilon = \varepsilon_2 \mathbin{\&} \text{dom}(\varepsilon_1) \\ \mathbf{error} & \text{otherwise} \end{cases} \\
(\text{Rpair}) \quad \langle \varepsilon_1 \mathbf{u}_1 :: \mathbf{G}_1, \varepsilon_2 \mathbf{u}_2 :: \mathbf{G}_2 \rangle \longrightarrow (\varepsilon_1 \times \varepsilon_2) \langle \mathbf{u}_1, \mathbf{u}_2 \rangle :: \mathbf{G}_1 \times \mathbf{G}_2 \\
(\text{Rproj}_i) \quad \pi_i(\varepsilon \langle \mathbf{u}_1, \mathbf{u}_2 \rangle :: \mathbf{G}_1 \times \mathbf{G}_2) \longrightarrow \mathbf{p}_i(\varepsilon) \mathbf{u}_i :: \mathbf{G}_i \\
(\text{RappG}) \quad (\varepsilon (\Lambda \mathbf{X}. \mathbf{t}) :: \forall \mathbf{X}. \mathbf{G}) [\mathbf{F}] \longrightarrow (\text{schm}(\varepsilon) \mathbf{t} :: \mathbf{G})[\mathbf{X} \mapsto \mathbf{F}]
\end{array}$$

$\mathbf{t} \mapsto \mathbf{t}$ or error Evaluation frames and reduction

$$\begin{array}{c}
\mathbf{f} ::= \varepsilon \square :: \mathbf{G} \mid \square \mathbf{t} \mid \mathbf{v} \square \mid \langle \square, \mathbf{t} \rangle \mid \langle \mathbf{v}, \square \rangle \mid \pi_i(\square) \mid \square [\mathbf{F}] \\
\text{R}_{\rightarrow} \frac{\mathbf{t} \longrightarrow \mathbf{t}'}{\mathbf{t} \mapsto \mathbf{t}'} \qquad \text{R}_{\mathbf{f}} \frac{\mathbf{t} \mapsto \mathbf{t}'}{\mathbf{f}[\mathbf{t}] \mapsto \mathbf{f}[\mathbf{t}']} \qquad \text{R}_{\text{err}} \frac{\mathbf{t} \longrightarrow \mathbf{error}}{\mathbf{t} \mapsto \mathbf{error}} \qquad \text{R}_{\mathbf{ferr}} \frac{\mathbf{t} \mapsto \mathbf{error}}{\mathbf{f}[\mathbf{t}] \mapsto \mathbf{error}}
\end{array}$$

Figure B.1: $F_{\varepsilon}^?$: Syntax, Static and Dynamic Semantics.

Definition B.2 (Well-formedness of δ)

$$\frac{}{\Delta \vdash \cdot} \qquad \frac{\Delta \vdash \delta \quad \Delta \vdash \mathbf{F}}{\Delta \vdash \delta, \mathbf{X} : \mathbf{F}}$$

Definition B.3 (Well-formedness of Γ)

$$\frac{}{\Delta \vdash \cdot} \qquad \frac{\Delta \vdash \Gamma \quad \Delta \vdash \mathbf{G}}{\Delta \vdash \Gamma, \mathbf{x} : \mathbf{G}}$$

Definition B.4 (Well-formedness of \mathbf{F} with respect to δ)

$$\frac{}{\delta \vdash \mathbf{B}} \qquad \frac{\mathbf{X} : \mathbf{X} \in \delta}{\delta \vdash \mathbf{X}}$$

Lemma B.5 (Canonical forms) *Consider a value $\Delta; \Gamma \vdash \mathbf{v} : \mathbf{G}$. Then $\mathbf{v} = \varepsilon \mathbf{u} :: \mathbf{G}$, with $\Delta; \Gamma \vdash \mathbf{u} : \mathbf{G}'$ and $\varepsilon : \mathbf{G}' \sim \mathbf{G}$. Furthermore:*

1. *If $\mathbf{G} = \mathbf{B}$, then $\mathbf{v} = \varepsilon_{\mathbf{B}} \mathbf{b} :: \mathbf{B}$, with $\Delta; \Gamma \vdash \mathbf{b} : \mathbf{B}$ and $\varepsilon_{\mathbf{B}} : \mathbf{B} \sim \mathbf{B}$.*
2. *If $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$, then $\mathbf{v} = \varepsilon (\lambda \mathbf{x} : \mathbf{G}'_1. \mathbf{t}) :: \mathbf{G}_1 \rightarrow \mathbf{G}_2$, with $\Delta; \Gamma, \mathbf{x} : \mathbf{G}'_1 \vdash \mathbf{t} : \mathbf{G}'_2$ and $\varepsilon : \mathbf{G}'_1 \rightarrow \mathbf{G}'_2 \sim \mathbf{G}_1 \rightarrow \mathbf{G}_2$.*
3. *If $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}_1$, then $\mathbf{v} = \varepsilon (\lambda \mathbf{X}. \mathbf{t}) :: \forall \mathbf{X}. \mathbf{G}_1$, with $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t} : \mathbf{G}'_1$ and $\varepsilon : \forall \mathbf{X}. \mathbf{G}'_1 \sim \forall \mathbf{X}. \mathbf{G}_1$.*
4. *If $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$, then $\mathbf{v} = \varepsilon \langle \mathbf{u}_1, \mathbf{u}_2 \rangle :: \mathbf{G}_1 \times \mathbf{G}_2$, with $\Delta; \Gamma \vdash \mathbf{u}_1 : \mathbf{G}'_1$, $\Delta; \Gamma \vdash \mathbf{u}_2 : \mathbf{G}'_2$ and $\varepsilon : \mathbf{G}'_1 \times \mathbf{G}'_2 \sim \mathbf{G}_1 \times \mathbf{G}_2$.*

PROOF. By direct inspection of the formation rules of evidence augmented terms. \square

Lemma B.6 *If $\varepsilon_1 : \mathbf{G}'_1 \sim \mathbf{G}_1$ and $\varepsilon_2 : \mathbf{G}'_2 \sim \mathbf{G}_2$, then $\varepsilon_1 \times \varepsilon_2 : \mathbf{G}'_1 \times \mathbf{G}'_2 \sim \mathbf{G}_1 \times \mathbf{G}_2$.*

PROOF. By definition of the judgment $\varepsilon_1 \times \varepsilon_2 : \mathbf{G}'_1 \times \mathbf{G}'_2 \sim \mathbf{G}_1 \times \mathbf{G}_2$ and the definition of evidences. \square

Lemma B.7 *If $\varepsilon : \mathbf{G}'_1 \times \mathbf{G}'_2 \sim \mathbf{G}_1 \times \mathbf{G}_2$ then $\mathbf{p}_i(\varepsilon) : \mathbf{G}'_i \sim \mathbf{G}_i$.*

PROOF. By definition of judgment $\mathbf{p}_i(\varepsilon) : \mathbf{G}'_i \sim \mathbf{G}_i$ and the definition of evidences. \square

Lemma B.8 *If $\varepsilon : \mathbf{G}'_1 \rightarrow \mathbf{G}'_2 \sim \mathbf{G}_1 \rightarrow \mathbf{G}_2$ then $\text{dom}(\varepsilon) : \mathbf{G}_1 \sim \mathbf{G}'_1$ and $\text{cod}(\varepsilon) : \mathbf{G}'_2 \sim \mathbf{G}_2$.*

PROOF. By definition of judgments $\text{dom}(\varepsilon) : \mathbf{G}_1 \sim \mathbf{G}'_1$ and $\text{cod}(\varepsilon) : \mathbf{G}'_2 \sim \mathbf{G}_2$, and the definition of evidences. \square

Lemma B.9 *If $\varepsilon : \forall X. G' \sim \forall X. G$ then $\text{schm}(\varepsilon) : G' \sim G$.*

PROOF. By definition of judgments $\text{schm}(\varepsilon) : G' \sim G$. □

[Progress] If $\vdash t : G$ then t is a value v , or $t \mapsto t'$, or $t \mapsto \mathbf{error}$.

PROOF. By straightforward induction on typing derivations. The variable case cannot occur (because t is closed). The cases for lambda-abstractions, type-abstraction, base values and pairs are immediate, since they are values. The remaining cases are more interesting.

Case. $t = t_1 t_2$, $\vdash t_1 : G_{11} \rightarrow G_{12}$, $\vdash t_2 : G_{11}$ and $G = G_{12}$. By the induction hypothesis, either t_1 is a value, it reduces to error ($t_1 \mapsto \mathbf{error}$) or it can make a step of evaluation ($t_1 \mapsto t'_1$); likewise t_2 . If t_1 can take a step, then rule **Rf** applies to t . If t_1 reduces to \mathbf{error} , then rule **Rferr** applies to t . If t_1 is a value and t_2 can take a step, then rule **Rf** applies. If t_1 is a value and t_2 reduces to \mathbf{error} , then rule **Rferr** applies. Finally, if both t_1 and t_2 are values, then the canonical forms Lemma B.5 tells us that t_1 has the form $\varepsilon (\lambda x : G'_1. t''_1) :: G_{11} \rightarrow G_{12}$ and t_2 has the form $\varepsilon' u :: G_{11}$; so rule (**Rapp**) applies to t . Therefore, $t_1 t_2 \rightarrow t''$, for some t'' , or $t_1 t_2 \rightarrow \mathbf{error}$; so rule **R \rightarrow** or **Rerr** applies to t , respectively.

Case. $t = \langle t_1, t_2 \rangle$, $\vdash t_1 : G_1$, $\vdash t_2 : G_2$ and $G = G_1 \times G_2$. By the induction hypothesis, either t_1 is a value, it reduces to error ($t_1 \mapsto \mathbf{error}$) or it can make a step of evaluation ($t_1 \mapsto t'_1$); likewise t_2 . If t_1 can take a step, then rule **Rf** applies to t . If t_1 reduces to \mathbf{error} , then rule **Rferr** applies to t . If t_1 is a value and t_2 can take a step, then rule **Rf** applies. If t_1 is a value and t_2 reduces to \mathbf{error} , then rule **Rferr** applies. Finally, if both t_1 and t_2 are values, then the canonical forms Lemma B.5 tells us that t_1 has the form $\varepsilon'_1 u_1 :: G_1$ and t_2 has the form $\varepsilon'_2 u_2 :: G_2$; so rule (**Rpair**) applies to t . Therefore, $\langle t_1, t_2 \rangle \rightarrow t''$, for some t'' , so rule **R \rightarrow** applies to t .

Case. $t = \varepsilon t_1 :: G$ and $\vdash t_1 : G'$. By the induction hypothesis, either t_1 is a value, it reduces to error ($t_1 \mapsto \mathbf{error}$) or it can make a step of evaluation ($t_1 \mapsto t'_1$). If t_1 can take a step, then rule **Rf** applies to t . If t_1 reduces to \mathbf{error} , then rule **Rferr** applies to t . Finally, if t_1 is a value, then the canonical forms Lemma B.5 tells us that t_1 has the form $\varepsilon' u :: G'$; so rule (**Rasc**) applies to t . Therefore, $\varepsilon t_1 :: G \rightarrow t''$, for some t'' , or $\varepsilon t_1 :: G \rightarrow \mathbf{error}$; so rule **R \rightarrow** or **Rerr** applies to t , respectively.

Case. $t = \pi_i(t_1)$, $\vdash t_1 : G_1 \times G_2$ and $G = G_i$. By the induction hypothesis, either t_1 is a value, it reduces to error ($t_1 \mapsto \mathbf{error}$) or it can make a step of evaluation ($t_1 \mapsto t'_1$). If t_1 can take a step, then rule **Rf** applies to t . If t_1 reduces to \mathbf{error} , then rule **Rferr** applies to t . Finally, if t_1 is a value, then the canonical forms Lemma B.5 tells us that t_1 has the form $\varepsilon \langle u_1, u_2 \rangle :: G_1 \times G_2$; so rule (**Rproji**) applies to t . Therefore, $\pi_i(t_1) \rightarrow t''$, for some t'' ; so rule **R \rightarrow** applies to t .

Case. $t = t_1 [F]$, $\vdash t_1 : \forall X. G_1$ and $G = G_1[X \mapsto F]$. By the induction hypothesis, either t_1 is a value, it reduces to error ($t_1 \mapsto \mathbf{error}$) or it can make a step of evaluation ($t_1 \mapsto t'_1$). If t_1 can take a step, then rule **Rf** applies to t . If t_1 reduces to \mathbf{error} , then rule **Rferr** applies to t . Finally, if t_1 is a value, then the canonical forms Lemma B.5 tells us that t_1 has the form $\varepsilon \lambda X. t''_1 :: \forall X. G_1$; so rule (**RappG**) applies to t . Therefore, $t_1 [F] \rightarrow t''$, for some t'' ; so rule **R \rightarrow** applies to t . □

[Preservation \rightarrow] If $\vdash t : G$ and $t \rightarrow t'$ then $\vdash t' : G$.

PROOF. By straightforward induction on typing derivations. The variable case cannot occur (because \mathbf{t} is closed). The cases for lambda-abstractions, type-abstraction, base values and pairs cannot also occur, since they are values. The remaining cases are more interesting.

Case. $\mathbf{t} = \mathbf{t}_1 \mathbf{t}_2$, $\vdash \mathbf{t}_1 : \mathbf{G}_{11} \rightarrow \mathbf{G}_{12}$, $\vdash \mathbf{t}_2 : \mathbf{G}_{11}$ and $\mathbf{G} = \mathbf{G}_{12}$. From the evaluation rules, we see that there is only one rule by which $\mathbf{t} \rightarrow \mathbf{t}'$ can be derived: (Rapp). Therefore, \mathbf{t}_1 has the form $\varepsilon(\lambda \mathbf{x} : \mathbf{G}'_1. \mathbf{t}'_1) :: \mathbf{G}_{11} \rightarrow \mathbf{G}_{12}$, \mathbf{t}_2 has the form $\varepsilon' \mathbf{u} :: \mathbf{G}_{11}$ and $\mathbf{t}' = \text{cod}(\varepsilon)(\mathbf{t}'_1[(\varepsilon' \ ; \ \text{dom}(\varepsilon) \ \mathbf{u} :: \mathbf{G}'_1)/\mathbf{x}]) :: \mathbf{G}_{12}$. By Lemma B.8, we get that $\text{dom}(\varepsilon) : \mathbf{G}_{11} \sim \mathbf{G}'_1$ and $\text{cod}(\varepsilon) : \mathbf{G}'_2 \sim \mathbf{G}_{12}$. Let us define $\mathbf{v} = (\varepsilon' \ ; \ \text{dom}(\varepsilon) \ \mathbf{u} :: \mathbf{G}'_1)$. Since $\vdash \mathbf{u} : \mathbf{G}'_1$ and $\varepsilon' \ ; \ \text{dom}(\varepsilon) : \mathbf{G}'_1 \sim \mathbf{G}'_1$, by rule \mathbf{G}_{asc} , we get that $\vdash \mathbf{v} : \mathbf{G}'_1$. Since $\vdash \mathbf{v} : \mathbf{G}'_1$ and $\mathbf{x} : \mathbf{G}'_1 \vdash \mathbf{t}'_1 : \mathbf{G}_{12}$, then $\vdash \mathbf{t}'_1[\mathbf{v}/\mathbf{x}] : \mathbf{G}_{12}$. Since $\text{cod}(\varepsilon) : \mathbf{G}'_2 \sim \mathbf{G}_{12}$, by rule \mathbf{G}_{asc} , we get that $\vdash \mathbf{t}' : \mathbf{G}_{12}$. Therefore, the result holds.

Case. $\mathbf{t} = \langle \mathbf{t}_1, \mathbf{t}_2 \rangle$, $\vdash \mathbf{t}_1 : \mathbf{G}_1$, $\vdash \mathbf{t}_2 : \mathbf{G}_2$ and $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$. From the evaluation rules, we see that there is only one rule by which $\mathbf{t} \rightarrow \mathbf{t}'$ can be derived: (Rpair). Therefore, \mathbf{t}_1 has the form $\varepsilon_1 \mathbf{u}_1 :: \mathbf{G}_1$, \mathbf{t}_2 has the form $\varepsilon_2 \mathbf{u}_2 :: \mathbf{G}_2$ and $\mathbf{t}' = (\varepsilon_1 \times \varepsilon_2) \langle \mathbf{u}_1, \mathbf{u}_2 \rangle :: \mathbf{G}_1 \times \mathbf{G}_2$. By Lemma B.6, we get that $\varepsilon_1 \times \varepsilon_2 : \mathbf{G}'_1 \times \mathbf{G}'_2 \sim \mathbf{G}_1 \times \mathbf{G}_2$. By rules \mathbf{G}_{asc} and \mathbf{G}_\times , we get that $\vdash \mathbf{t}' : \mathbf{G}_1 \times \mathbf{G}_2$. Therefore, the result holds.

Case. $\mathbf{t} = \varepsilon \mathbf{t}_1 :: \mathbf{G}$ and $\vdash \mathbf{t}_1 : \mathbf{G}'$. From the evaluation rules, we see that there is only one rule by which $\mathbf{t} \rightarrow \mathbf{t}'$ can be derived: (Rasc). Therefore, \mathbf{t}_1 has the form $\varepsilon' \mathbf{u} :: \mathbf{G}'$ and $\mathbf{t}' = (\varepsilon' \ ; \ \varepsilon) \mathbf{u} :: \mathbf{G}$. Since $\varepsilon' \ ; \ \varepsilon : \mathbf{G}' \sim \mathbf{G}$, by rule \mathbf{G}_{asc} , we get that $\vdash \mathbf{t}' : \mathbf{G}$. Therefore, the result holds.

Case. $\mathbf{t} = \pi_i(\mathbf{t}_1)$, $\vdash \mathbf{t}_1 : \mathbf{G}_1 \times \mathbf{G}_2$ and $\mathbf{G} = \mathbf{G}_i$. From the evaluation rules, we see that there is only one rule by which $\mathbf{t} \rightarrow \mathbf{t}'$ can be derived: (Rproji). Therefore, \mathbf{t}_1 has the form $\varepsilon \langle \mathbf{u}_1, \mathbf{u}_2 \rangle :: \mathbf{G}_1 \times \mathbf{G}_2$ and $\mathbf{t}' = \mathbf{p}_i(\varepsilon) \mathbf{u}_i :: \mathbf{G}_i$. By Lemma B.7, we get that $\mathbf{p}_i(\varepsilon) : \mathbf{G}'_i \sim \mathbf{G}_i$. Since $\mathbf{p}_i(\varepsilon) : \mathbf{G}'_i \sim \mathbf{G}_i$, by rule \mathbf{G}_{asc} , we get that $\vdash \mathbf{t}' : \mathbf{G}$. Therefore, the result holds.

Case. $\mathbf{t} = \mathbf{t}_1 [\mathbf{F}]$, $\vdash \mathbf{t}_1 : \forall \mathbf{X}. \mathbf{G}_1$ and $\mathbf{G} = \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}]$. From the evaluation rules, we see that there is only one rule by which $\mathbf{t} \rightarrow \mathbf{t}'$ can be derived: (RappG). Therefore, \mathbf{t}_1 has the form $\varepsilon \lambda \mathbf{X}. \mathbf{t}'_1 :: \forall \mathbf{X}. \mathbf{G}_1$ and $\mathbf{t}' = (\text{schm}(\varepsilon)[\mathbf{X} \mapsto \mathbf{F}]) \mathbf{t}'_1[\mathbf{X} \mapsto \mathbf{F}] :: \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}]$. By Lemma B.9, we get that $\text{schm}(\varepsilon) : \mathbf{G}'_1 \sim \mathbf{G}_1$. By Lemma B.55, we get that $\Delta; \Gamma \vdash \mathbf{t}'_1[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}'_1[\mathbf{X} \mapsto \mathbf{F}]$. By Lemma B.58, we get that $\text{schm}(\varepsilon)[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}'_1[\mathbf{X} \mapsto \mathbf{F}] \sim \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}]$. By rule \mathbf{G}_{asc} , we get that $\vdash \mathbf{t}' : \mathbf{G}$. Therefore, the result holds. □

[Preservation] If $\vdash \mathbf{t} : \mathbf{G}$ and $\mathbf{t} \mapsto \mathbf{t}'$ then $\vdash \mathbf{t}' : \mathbf{G}$.

PROOF. By straightforward induction on $\mathbf{t} \mapsto \mathbf{t}'$. The **Rerr** and **Rferr** cases cannot occur because **error** is not a term. The remaining cases are more interesting.

Case (R \rightarrow). $\mathbf{t} \rightarrow \mathbf{t}'$. Directly by B.1.

Case (Rf). $\mathbf{t} = \mathbf{f}[\mathbf{t}_1]$ and $\mathbf{t}_1 \mapsto \mathbf{t}'_1$. We know that $\vdash \mathbf{f}[\mathbf{t}_1] : \mathbf{G}$ and $\vdash \mathbf{t}_1 : \mathbf{G}'$, where $\mathbf{f} : \mathbf{G}' \rightarrow \mathbf{G}$. Then, by the induction hypothesis on $\vdash \mathbf{t}_1 : \mathbf{G}'$, we get that $\vdash \mathbf{t}'_1 : \mathbf{G}$. Therefore, $\vdash \mathbf{f}[\mathbf{t}'_1] : \mathbf{G}$ and the result holds. □

Lemma B.10 (Type Safety) *If* $\vdash \mathbf{t} : \mathbf{G}$ *then either* $\mathbf{t} \xrightarrow{*} \mathbf{v}$ *with* $\vdash \mathbf{v} : \mathbf{G}$, $\mathbf{t} \xrightarrow{*} \mathbf{error}$, *or* \mathbf{t} *diverges.*

PROOF. Direct by B.1 and B.1. □

B.2 Evidence in $F_\varepsilon^?$

This section presents definitions and proof related to the evidence.

B.2.1 Proof Relevant Gradual Type Precision

Lemma B.11 (Inversion) *Suppose $\mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$.*

1. If $\mathbf{G}_2 = \mathbf{B}$, then $\mathbf{G}_1 = \mathbf{B}$ and $\mathbf{c} = \mathbf{B}$.
2. If $\mathbf{G}_2 = \mathbf{X}$, then $\mathbf{G}_1 = \mathbf{X}$ and $\mathbf{c} = \mathbf{X}$.
3. If $\mathbf{G}_2 = \mathbf{G}'_2 \rightarrow \mathbf{G}''_2$, then $\mathbf{G}_1 = \mathbf{G}'_1 \rightarrow \mathbf{G}''_1$ and $\mathbf{c} = \mathbf{c}' \rightarrow \mathbf{c}''$.
4. If $\mathbf{G}_2 = \mathbf{G}'_2 \times \mathbf{G}''_2$, then $\mathbf{G}_1 = \mathbf{G}'_1 \times \mathbf{G}''_1$ and $\mathbf{c} = \mathbf{c}' \times \mathbf{c}''$.
5. If $\mathbf{G}_2 = \forall \mathbf{X}. \mathbf{G}'_2$, then $\mathbf{G}_1 = \forall \mathbf{X}. \mathbf{G}'_1$ and $\mathbf{c} = \forall \mathbf{X}. \mathbf{c}'$.
6. If $\mathbf{G}_2 = ?_\delta$, then one of the following hold:
 - $\mathbf{G}_1 = \mathbf{B}$ and $\mathbf{c} = \text{inj}_{\mathbf{B}}$.
 - $\mathbf{G}_1 = \mathbf{X}$ and $\mathbf{c} = \text{inj}_{\mathbf{X}}$ and $\mathbf{X} : \mathbf{X} \in \delta$.
 - $\mathbf{G}_1 = \mathbf{B}$, $\mathbf{c} = \text{inj}_{\mathbf{X}}$ and $\mathbf{X} : \mathbf{B} \in \delta$.
 - $\mathbf{G}_1 = \mathbf{Y}$, $\mathbf{c} = \text{inj}_{\mathbf{X}}$, $\mathbf{X} \neq \mathbf{Y}$, $\mathbf{X} : \mathbf{Y} \in \delta$ and $\mathbf{Y} : \mathbf{Y} \in \delta$.
 - $\mathbf{G}_1 = \mathbf{G}'_1 \rightarrow \mathbf{G}''_1$ and $\mathbf{c} = \text{inj}_{\rightarrow}(\mathbf{c}' \rightarrow \mathbf{c}'')$.
 - $\mathbf{G}_1 = \mathbf{G}'_1 \times \mathbf{G}''_1$ and $\mathbf{c} = \text{inj}_{\times}(\mathbf{c}' \times \mathbf{c}'')$.
 - $\mathbf{G}_1 = \forall \mathbf{X}. \mathbf{G}'_1$ and $\mathbf{c} = \text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}')$.
 - $\mathbf{G}_1 = ?_{\delta'}$, $\mathbf{c} = \text{inj}_{?}$ and $\delta' \subseteq \delta$.

PROOF. Straightforward by Definition of $\mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$. □

Lemma B.12 (Inversion of the type precision transitivity) *Suppose $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}_3$, $\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$ and $\mathbf{c}_1; \mathbf{c}_2 = \mathbf{c}_3$.*

1. If $\mathbf{c}_3 = \mathbf{B}$, then $\mathbf{G}_1 = \mathbf{G}_2 = \mathbf{G}_3 = \mathbf{B}$ and $\mathbf{c}_1 = \mathbf{c}_2 = \mathbf{B}$.
2. If $\mathbf{c}_3 = \mathbf{X}$, then $\mathbf{G}_1 = \mathbf{G}_2 = \mathbf{G}_3 = \mathbf{X}$ and $\mathbf{c}_1 = \mathbf{c}_2 = \mathbf{X}$.
3. If $\mathbf{c}_3 = \forall \mathbf{X}. \mathbf{c}'_3$, then $\mathbf{G}_1 = \forall \mathbf{X}. \mathbf{G}'_1$, $\mathbf{G}_2 = \forall \mathbf{X}. \mathbf{G}'_2$, $\mathbf{G}_3 = \forall \mathbf{X}. \mathbf{G}'_3$, $\mathbf{c}_1 = \forall \mathbf{X}. \mathbf{c}'_1$, $\mathbf{c}_2 = \forall \mathbf{X}. \mathbf{c}'_2$ and $\mathbf{c}'_1; \mathbf{c}'_2 = \mathbf{c}'_3$.
4. If $\mathbf{c}_3 = \mathbf{c}'_3 \rightarrow \mathbf{c}''_3$, then $\mathbf{G}_1 = \mathbf{G}'_1 \rightarrow \mathbf{G}''_1$, $\mathbf{G}_2 = \mathbf{G}'_2 \rightarrow \mathbf{G}''_2$, $\mathbf{G}_3 = \mathbf{G}'_3 \rightarrow \mathbf{G}''_3$, $\mathbf{c}_1 = \mathbf{c}'_1 \rightarrow \mathbf{c}''_1$, $\mathbf{c}_2 = \mathbf{c}'_2 \rightarrow \mathbf{c}''_2$, $\mathbf{c}'_1; \mathbf{c}'_2 = \mathbf{c}'_3$ and $\mathbf{c}''_1; \mathbf{c}''_2 = \mathbf{c}''_3$.
5. If $\mathbf{c}_3 = \text{inj}_{\mathbf{B}}$, then one of the following hold:
 - $\mathbf{G}_1 = \mathbf{G}_2 = \mathbf{B}$, $\mathbf{G}_3 = ?_\delta$, $\mathbf{c}_1 = \mathbf{B}$ and $\mathbf{c}_2 = \text{inj}_{\mathbf{B}}$, or

- $\mathbf{G}_1 = \mathbf{B}$, $\mathbf{G}_2 = ?_\delta$, $\mathbf{G}_3 = ?_{\delta'}$, $\delta \subseteq \delta'$, $\mathbf{c}_1 = \text{inj}_{\mathbf{B}}$ and $\mathbf{c}_2 = \text{inj}_{?}$.
6. If $\mathbf{c}_3 = \text{inj}_{\mathbf{X}}$, then one of the following hold:
- $\mathbf{G}_1 = \mathbf{G}_2 = \mathbf{F}$, $\mathbf{G}_3 = ?_\delta$, $\mathbf{X} : \mathbf{F} \in \delta$, $\mathbf{c}_1 = \mathbf{F}$ and $\mathbf{c}_2 = \text{inj}_{\mathbf{X}}$, or
 - $\mathbf{G}_1 = \mathbf{F}$, $\mathbf{G}_2 = ?_\delta$, $\mathbf{X} : \mathbf{F} \in \delta$, $\mathbf{G}_3 = ?_{\delta'}$, $\delta \subseteq \delta'$, $\mathbf{c}_1 = \text{inj}_{\mathbf{X}}$ and $\mathbf{c}_2 = \text{inj}_{?}$.
7. If $\mathbf{c}_3 = \text{inj}_{\forall}(\mathbf{c})$, then $\mathbf{c} = \forall \mathbf{X}. \mathbf{c}'_3$ and one of the following hold:
- $\mathbf{G}_1 = \forall \mathbf{X}. \mathbf{G}'_1$, $\mathbf{G}_2 = \forall \mathbf{X}. \mathbf{G}'_2$, $\mathbf{G}_3 = ?_\delta$, $\mathbf{c}_1 = \forall \mathbf{X}. \mathbf{c}'_1$, $\mathbf{c}_2 = \text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}'_2)$ and $\mathbf{c}'_1; \mathbf{c}'_2 = \mathbf{c}'_3$, or
 - $\mathbf{G}_1 = \forall \mathbf{X}. \mathbf{G}'_1$, $\mathbf{G}_2 = ?_\delta$, $\mathbf{G}_3 = ?_{\delta'}$, $\delta \subseteq \delta'$, $\mathbf{c}_1 = \text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}'_3)$ and $\mathbf{c}_2 = \text{inj}_{?}$.
8. If $\mathbf{c}_3 = \text{inj}_{\rightarrow}(\mathbf{c})$, then $\mathbf{c} = \mathbf{c}'_3 \rightarrow \mathbf{c}''_3$ and one of the following hold:
- $\mathbf{G}_1 = \mathbf{G}'_1 \rightarrow \mathbf{G}''_1$, $\mathbf{G}_2 = \mathbf{G}'_2 \rightarrow \mathbf{G}''_2$, $\mathbf{G}_3 = ?_\delta$, $\mathbf{c}_1 = \mathbf{c}'_1 \rightarrow \mathbf{c}''_1$, $\mathbf{c}_2 = \text{inj}_{\rightarrow}(\mathbf{c}'_2 \rightarrow \mathbf{c}''_2)$, $\mathbf{c}'_1; \mathbf{c}'_2 = \mathbf{c}'_3$ and $\mathbf{c}''_1; \mathbf{c}''_2 = \mathbf{c}''_3$, or
 - $\mathbf{G}_1 = \mathbf{G}'_1 \rightarrow \mathbf{G}''_1$, $\mathbf{G}_2 = ?_\delta$, $\mathbf{G}_3 = ?_{\delta'}$, $\delta \subseteq \delta'$, $\mathbf{c}_1 = \text{inj}_{\rightarrow}(\mathbf{c}'_3 \rightarrow \mathbf{c}''_3)$ and $\mathbf{c}_2 = \text{inj}_{?}$.

PROOF. Straightforward by Definition of $\mathbf{c}_1; \mathbf{c}_2 = \mathbf{c}_3$. □

Lemma B.13 (Associativity of Precision Transitivity) *For all \mathbf{c}_4 we get that $\mathbf{c}_1; (\mathbf{c}_2; \mathbf{c}_3) = \mathbf{c}_4$ iff $(\mathbf{c}_1; \mathbf{c}_2); \mathbf{c}_3 = \mathbf{c}_4$.*

PROOF. We proceed by induction on \mathbf{c}_4 .

- If $\mathbf{c}_4 = \mathbf{B}$, then by Lemma B.12, we know that $\mathbf{c}_1 = \mathbf{B}$, $\mathbf{c}_2 = \mathbf{B}$ and $\mathbf{c}_3 = \mathbf{B}$. Thus, we know that $(\mathbf{B}; \mathbf{B}); \mathbf{B} = \mathbf{B}$ and $\mathbf{B}; (\mathbf{B}; \mathbf{B}) = \mathbf{B}$. Therefore, the result holds.
- If $\mathbf{c}_4 = \mathbf{X}$, then by Lemma B.12, we know that $\mathbf{c}_1 = \mathbf{X}$, $\mathbf{c}_2 = \mathbf{X}$ and $\mathbf{c}_3 = \mathbf{X}$. Thus, we know that $(\mathbf{X}; \mathbf{X}); \mathbf{X} = \mathbf{X}$ and $\mathbf{X}; (\mathbf{X}; \mathbf{X}) = \mathbf{X}$. Therefore, the result holds.
- If $\mathbf{c}_4 = \forall \mathbf{X}. \mathbf{c}'_4$, then by Lemma B.12, we know that $\mathbf{c}_1 = \forall \mathbf{X}. \mathbf{c}_1$, $\mathbf{c}_2 = \forall \mathbf{X}. \mathbf{c}'_2$ and $\mathbf{c}_3 = \forall \mathbf{X}. \mathbf{c}'_3$. If $(\forall \mathbf{X}. \mathbf{c}_1; \forall \mathbf{X}. \mathbf{c}'_2); \forall \mathbf{X}. \mathbf{c}'_3 = \forall \mathbf{X}. \mathbf{c}'_4$, we know by inspecting the rules of gradual type precision transitivity that

- there exists \mathbf{c} such that $\forall \mathbf{X}. \mathbf{c}_1; \forall \mathbf{X}. \mathbf{c}'_2 = \mathbf{c}$, by Definition B.21.
- $\mathbf{c} = \forall \mathbf{X}. \mathbf{c}'$ and $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'$.
- $\forall \mathbf{X}. \mathbf{c}'; \forall \mathbf{X}. \mathbf{c}'_3 = \forall \mathbf{X}. \mathbf{c}'_4$ and $\mathbf{c}'; \mathbf{c}'_3 = \mathbf{c}'_4$.
- there exists \mathbf{c}'' such that $\forall \mathbf{X}. \mathbf{c}'_2; \forall \mathbf{X}. \mathbf{c}'_3 = \mathbf{c}''$, by Definition B.21.
- $\mathbf{c}'' = \forall \mathbf{X}. \mathbf{c}'''$ and $\mathbf{c}'_2; \mathbf{c}'_3 = \mathbf{c}'''$.
- By the induction hypothesis on \mathbf{c}'_4 , with $(\mathbf{c}_1; \mathbf{c}'_2); \mathbf{c}'_3 = \mathbf{c}'_4$, we know that $\mathbf{c}_1; (\mathbf{c}'_2; \mathbf{c}'_3) = \mathbf{c}'_4$.
- Thus, we know that $\forall \mathbf{X}. \mathbf{c}_1; (\forall \mathbf{X}. \mathbf{c}'_2; \forall \mathbf{X}. \mathbf{c}'_3) = \forall \mathbf{X}. \mathbf{c}'_4$.

In a similar way, if $\forall \mathbf{X}. \mathbf{c}_1; (\forall \mathbf{X}. \mathbf{c}'_2; \forall \mathbf{X}. \mathbf{c}'_3) = \forall \mathbf{X}. \mathbf{c}'_4$, we know by inspecting the rules of gradual type precision transitivity that

- there exists \mathbf{c} such that $\forall \mathbf{X}. \mathbf{c}'_2; \forall \mathbf{X}. \mathbf{c}'_3 = \mathbf{c}$, by Definition B.21.

- $\mathbf{c} = \forall \mathbf{X}.\mathbf{c}'$ and $\mathbf{c}'_2;\mathbf{c}'_3 = \mathbf{c}'$.
- $\forall \mathbf{X}.\mathbf{c}_1;\forall \mathbf{X}.\mathbf{c}' = \forall \mathbf{X}.\mathbf{c}'_4$ and $\mathbf{c}_1;\mathbf{c}' = \mathbf{c}'_4$.
- there exists \mathbf{c}'' such that $\forall \mathbf{X}.\mathbf{c}_1;\forall \mathbf{X}.\mathbf{c}'_2 = \mathbf{c}''$, by Definition B.21.
- $\mathbf{c}'' = \forall \mathbf{X}.\mathbf{c}'''$ and $\mathbf{c}_1;\mathbf{c}'_2 = \mathbf{c}'''$.
- By the induction hypothesis on \mathbf{c}'_4 , with $\mathbf{c}_1;(\mathbf{c}'_2;\mathbf{c}'_3) = \mathbf{c}'_4$, we know that $(\mathbf{c}_1;\mathbf{c}'_2);\mathbf{c}'_3 = \mathbf{c}'_4$.
- Thus, we know that $(\forall \mathbf{X}.\mathbf{c}_1;\forall \mathbf{X}.\mathbf{c}'_2);\forall \mathbf{X}.\mathbf{c}'_3 = \forall \mathbf{X}.\mathbf{c}'_4$.

Therefore, the result holds.

- If $\mathbf{c}_4 = \mathbf{c}'_4 \rightarrow \mathbf{c}''_4$, then by Lemma B.12, we know that $\mathbf{c}_1 = \mathbf{c}_1 \rightarrow \mathbf{c}''_1$, $\mathbf{c}_2 = \mathbf{c}'_2 \rightarrow \mathbf{c}''_2$ and $\mathbf{c}_3 = \mathbf{c}'_3 \rightarrow \mathbf{c}''_3$. If $(\mathbf{c}_1 \rightarrow \mathbf{c}''_1;\mathbf{c}'_2 \rightarrow \mathbf{c}''_2);\mathbf{c}'_3 \rightarrow \mathbf{c}''_3 = \mathbf{c}'_4 \rightarrow \mathbf{c}''_4$, we know by inspecting the rules of gradual type precision transitivity that
 - there exists \mathbf{c} such that $\mathbf{c}_1 \rightarrow \mathbf{c}''_1;\mathbf{c}'_2 \rightarrow \mathbf{c}''_2 = \mathbf{c}$, by Definition B.21.
 - $\mathbf{c} = \mathbf{c}_{11} \rightarrow \mathbf{c}_{21}$, $\mathbf{c}_1;\mathbf{c}'_2 = \mathbf{c}_{11}$ and $\mathbf{c}''_1;\mathbf{c}''_2 = \mathbf{c}_{21}$.
 - $\mathbf{c}_{11} \rightarrow \mathbf{c}_{21};\mathbf{c}'_3 \rightarrow \mathbf{c}''_3 = \mathbf{c}'_4 \rightarrow \mathbf{c}''_4$, $\mathbf{c}_{11};\mathbf{c}'_3 = \mathbf{c}'_4$ and $\mathbf{c}_{21};\mathbf{c}''_3 = \mathbf{c}''_4$.
 - there exists \mathbf{c}' such that $\mathbf{c}'_2 \rightarrow \mathbf{c}''_2;\mathbf{c}'_3 \rightarrow \mathbf{c}''_3 = \mathbf{c}'$, by Definition B.21.
 - $\mathbf{c}' = \mathbf{c}_{12} \rightarrow \mathbf{c}_{22}$, $\mathbf{c}'_2;\mathbf{c}'_3 = \mathbf{c}_{12}$ and $\mathbf{c}''_2;\mathbf{c}''_3 = \mathbf{c}_{22}$.
 - By the induction hypothesis on \mathbf{c}'_4 and \mathbf{c}''_4 , with $(\mathbf{c}_1;\mathbf{c}'_2);\mathbf{c}'_3 = \mathbf{c}'_4$ and $(\mathbf{c}''_1;\mathbf{c}''_2);\mathbf{c}''_3 = \mathbf{c}''_4$, we know that $\mathbf{c}_1;(\mathbf{c}'_2;\mathbf{c}'_3) = \mathbf{c}'_4$ and $\mathbf{c}''_1;(\mathbf{c}''_2;\mathbf{c}''_3) = \mathbf{c}''_4$.
 - Thus, we know that $\mathbf{c}_1 \rightarrow \mathbf{c}''_1;(\mathbf{c}'_2 \rightarrow \mathbf{c}''_2;\mathbf{c}'_3 \rightarrow \mathbf{c}''_3) = \mathbf{c}'_4 \rightarrow \mathbf{c}''_4$.

In a similar way, if $\mathbf{c}_1 \rightarrow \mathbf{c}''_1;(\mathbf{c}'_2 \rightarrow \mathbf{c}''_2;\mathbf{c}'_3 \rightarrow \mathbf{c}''_3) = \mathbf{c}'_4 \rightarrow \mathbf{c}''_4$, we know by inspecting the rules of gradual type precision transitivity that

- there exists \mathbf{c} such that $\mathbf{c}'_2 \rightarrow \mathbf{c}''_2;\mathbf{c}'_3 \rightarrow \mathbf{c}''_3 = \mathbf{c}$, by Definition B.21.
- $\mathbf{c} = \mathbf{c}_{11} \rightarrow \mathbf{c}_{21}$, $\mathbf{c}'_2;\mathbf{c}'_3 = \mathbf{c}_{11}$ and $\mathbf{c}''_2;\mathbf{c}''_3 = \mathbf{c}_{21}$.
- $\mathbf{c}_1 \rightarrow \mathbf{c}''_1;\mathbf{c}_{11} \rightarrow \mathbf{c}_{21} = \mathbf{c}'_4 \rightarrow \mathbf{c}''_4$, $\mathbf{c}_1;\mathbf{c}_{11} = \mathbf{c}'_4$ and $\mathbf{c}''_1;\mathbf{c}_{21} = \mathbf{c}''_4$.
- there exists \mathbf{c}' such that $\mathbf{c}_1 \rightarrow \mathbf{c}''_1;\mathbf{c}'_2 \rightarrow \mathbf{c}''_2 = \mathbf{c}'$, by Definition B.21.
- $\mathbf{c}' = \mathbf{c}_{12} \rightarrow \mathbf{c}_{22}$, $\mathbf{c}_1;\mathbf{c}'_2 = \mathbf{c}_{12}$ and $\mathbf{c}''_1;\mathbf{c}''_2 = \mathbf{c}_{22}$.
- By the induction hypothesis on \mathbf{c}'_4 and \mathbf{c}''_4 , with $\mathbf{c}_1;(\mathbf{c}'_2;\mathbf{c}'_3) = \mathbf{c}'_4$ and $\mathbf{c}''_1;(\mathbf{c}''_2;\mathbf{c}''_3) = \mathbf{c}''_4$, we know that $(\mathbf{c}_1;\mathbf{c}'_2);\mathbf{c}'_3 = \mathbf{c}'_4$ and $(\mathbf{c}''_1;\mathbf{c}''_2);\mathbf{c}''_3 = \mathbf{c}''_4$.
- Thus, we know that $(\mathbf{c}_1 \rightarrow \mathbf{c}''_1;\mathbf{c}'_2 \rightarrow \mathbf{c}''_2);\mathbf{c}'_3 \rightarrow \mathbf{c}''_3 = \mathbf{c}'_4 \rightarrow \mathbf{c}''_4$.

Therefore, the result holds.

- If $\mathbf{c}_4 = \text{inj}_{\mathbf{B}}$, then by Lemma B.12, we have the following cases.
 - $\mathbf{c}_1 = \mathbf{B}$, $\mathbf{c}_2 = \mathbf{B}$ and $\mathbf{c}_3 = \text{inj}_{\mathbf{B}}$. Straightforward by looking at rules of gradual type precision transitivity ($\mathbf{B};\mathbf{B} = \mathbf{B}$ and $\mathbf{B};\text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$).
 - $\mathbf{c}_1 = \mathbf{B}$, $\mathbf{c}_2 = \text{inj}_{\mathbf{B}}$ and $\mathbf{c}_3 = \text{inj}_{\mathbf{?}}$. Straightforward by looking at rules of gradual type precision transitivity ($\mathbf{B};\text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$ and $\text{inj}_{\mathbf{B}};\text{inj}_{\mathbf{?}} = \text{inj}_{\mathbf{B}}$).
 - $\mathbf{c}_1 = \text{inj}_{\mathbf{B}}$, $\mathbf{c}_2 = \text{inj}_{\mathbf{?}}$ and $\mathbf{c}_3 = \text{inj}_{\mathbf{?}}$. Straightforward by looking at rules of gradual type precision transitivity ($\text{inj}_{\mathbf{B}};\text{inj}_{\mathbf{?}} = \text{inj}_{\mathbf{B}}$, $\text{inj}_{\mathbf{?}};\text{inj}_{\mathbf{?}} = \text{inj}_{\mathbf{?}}$ and $\text{inj}_{\mathbf{B}};\text{inj}_{\mathbf{?}} = \text{inj}_{\mathbf{B}}$).

- If $\mathbf{c}_4 = \mathbf{inj}_Y$, then by Lemma B.12, we have the following cases.
 - $\mathbf{c}_1 = \mathbf{F}$, $\mathbf{c}_2 = \mathbf{F}$ and $\mathbf{c}_3 = \mathbf{inj}_Y$. Straightforward by looking at rules of gradual type precision transitivity ($\mathbf{F};\mathbf{F} = \mathbf{F}$ and $\mathbf{F};\mathbf{inj}_Y = \mathbf{inj}_Y$).
 - $\mathbf{c}_1 = \mathbf{F}$, $\mathbf{c}_2 = \mathbf{inj}_Y$ and $\mathbf{c}_3 = \mathbf{inj}_?$. Then the result follows because $\mathbf{F};\mathbf{inj}_Y = \mathbf{inj}_Y$ and $\mathbf{inj}_Y;\mathbf{inj}_? = \mathbf{inj}_Y$, and therefore, $\mathbf{F};(\mathbf{inj}_Y;\mathbf{inj}_?) = \mathbf{inj}_Y$ and $(\mathbf{F};\mathbf{inj}_Y);\mathbf{inj}_? = \mathbf{inj}_Y$.
 - $\mathbf{c}_1 = \mathbf{inj}_Y$, $\mathbf{c}_2 = \mathbf{inj}_?$ and $\mathbf{c}_3 = \mathbf{inj}_?$. Then the result follows because $\mathbf{inj}_Y;\mathbf{inj}_? = \mathbf{inj}_Y$ and $\mathbf{inj}_?;\mathbf{inj}_? = \mathbf{inj}_?$, and therefore, $\mathbf{inj}_Y;(\mathbf{inj}_?;\mathbf{inj}_?) = \mathbf{inj}_Y$ and $(\mathbf{inj}_Y;\mathbf{inj}_?);\mathbf{inj}_? = \mathbf{inj}_Y$.
- If $\mathbf{c}_4 = \mathbf{inj}_\forall(\mathbf{c}'_4)$, then by Lemma B.12, we have the following cases.

- $\mathbf{c}_1 = \forall \mathbf{X}.\mathbf{c}_1$, $\mathbf{c}_2 = \forall \mathbf{X}.\mathbf{c}'_2$ and $\mathbf{c}_3 = \mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_3)$. If $(\forall \mathbf{X}.\mathbf{c}_1;\forall \mathbf{X}.\mathbf{c}'_2);\mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_3) = \forall \mathbf{X}.\mathbf{inj}_\forall(\mathbf{c}'_4)$, we know by inspecting the rules of gradual type precision transitivity that
 - * there exists \mathbf{c} such that $\forall \mathbf{X}.\mathbf{c}_1;\forall \mathbf{X}.\mathbf{c}'_2 = \mathbf{c}$, by Definition B.21.
 - * $\mathbf{c} = \forall \mathbf{X}.\mathbf{c}'$ and $\mathbf{c}_1;\mathbf{c}'_2 = \mathbf{c}'$.
 - * $\forall \mathbf{X}.\mathbf{c}';\mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_3) = \mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_4)$ and $\mathbf{c}';\mathbf{c}'_3 = \mathbf{c}'_4$.
 - * there exists \mathbf{c}'' such that $\forall \mathbf{X}.\mathbf{c}'_2;\mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_3) = \mathbf{c}''$, by Definition B.21.
 - * $\mathbf{c}'' = \mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}''')$ and $\mathbf{c}'_2;\mathbf{c}'_3 = \mathbf{c}'''$.
 - * By the induction hypothesis on \mathbf{c}'_4 , with $(\mathbf{c}_1;\mathbf{c}'_2);\mathbf{c}'_3 = \mathbf{c}'_4$, we know that $\mathbf{c}_1;(\mathbf{c}'_2;\mathbf{c}'_3) = \mathbf{c}'_4$.
 - * Thus, we know that $\forall \mathbf{X}.\mathbf{c}_1;(\forall \mathbf{X}.\mathbf{c}'_2;\mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_3)) = \mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_4)$.

In a similar way, if $\forall \mathbf{X}.\mathbf{c}_1;(\forall \mathbf{X}.\mathbf{c}'_2;\mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_3)) = \mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_4)$, we know by inspecting the rules of gradual type precision transitivity that

- * there exists \mathbf{c} such that $\forall \mathbf{X}.\mathbf{c}'_2;\mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_3) = \mathbf{c}$, by Definition B.21.
- * $\mathbf{c} = \mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}')$ and $\mathbf{c}'_2;\mathbf{c}'_3 = \mathbf{c}'$.
- * $\forall \mathbf{X}.\mathbf{c}_1;\mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}') = \mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_4)$ and $\mathbf{c}_1;\mathbf{c}' = \mathbf{c}'_4$.
- * there exists \mathbf{c}'' such that $\forall \mathbf{X}.\mathbf{c}_1;\forall \mathbf{X}.\mathbf{c}'_2 = \mathbf{c}''$, by Definition B.21.
- * $\mathbf{c}'' = \forall \mathbf{X}.\mathbf{c}'''$ and $\mathbf{c}_1;\mathbf{c}'_2 = \mathbf{c}'''$.
- * By the induction hypothesis on \mathbf{c}'_4 , with $\mathbf{c}_1;(\mathbf{c}'_2;\mathbf{c}'_3) = \mathbf{c}'_4$, we know that $(\mathbf{c}_1;\mathbf{c}'_2);\mathbf{c}'_3 = \mathbf{c}'_4$.
- * Thus, we know that $(\forall \mathbf{X}.\mathbf{c}_1;\forall \mathbf{X}.\mathbf{c}'_2);\mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_3) = \mathbf{inj}_\forall(\forall \mathbf{X}.\mathbf{c}'_4)$.

Therefore, the result holds.

- $\mathbf{c}_1 = \forall \mathbf{X}.\mathbf{c}_1$, $\mathbf{c}_2 = \mathbf{inj}_\forall(\mathbf{c}'_2)$ and $\mathbf{c}_3 = \mathbf{inj}_?$. Straightforward by looking at rules of gradual type precision transitivity ($\forall \mathbf{X}.\mathbf{c}_1;\mathbf{inj}_\forall(\mathbf{c}'_2) = \mathbf{inj}_\forall(\mathbf{c}'_4)$, $\mathbf{inj}_\forall(\mathbf{c}'_2);\mathbf{inj}_? = \mathbf{inj}_\forall(\mathbf{c}'_2)$ and $\mathbf{inj}_\forall(\mathbf{c}'_4);\mathbf{inj}_? = \mathbf{inj}_\forall(\mathbf{c}'_4)$).
- $\mathbf{c}_1 = \forall \mathbf{X}.\mathbf{c}_1$, $\mathbf{c}_2 = \mathbf{inj}_\forall(\mathbf{c}'_2)$ and $\mathbf{c}_3 = \mathbf{inj}_?$. Straightforward by looking at rules of gradual type precision transitivity ($\forall \mathbf{X}.\mathbf{c}_1;\mathbf{inj}_\forall(\mathbf{c}'_2) = \mathbf{inj}_\forall(\mathbf{c}'_4)$, $\mathbf{inj}_\forall(\mathbf{c}'_2);\mathbf{inj}_? = \mathbf{inj}_\forall(\mathbf{c}'_2)$ and $\mathbf{inj}_\forall(\mathbf{c}'_4);\mathbf{inj}_? = \mathbf{inj}_\forall(\mathbf{c}'_4)$).
- $\mathbf{c}_1 = \mathbf{inj}_\forall(\mathbf{c}_1)$, $\mathbf{c}_2 = \mathbf{inj}_?$ and $\mathbf{c}_3 = \mathbf{inj}_?$. Straightforward by looking at rules of gradual type precision transitivity ($\mathbf{inj}_\forall(\mathbf{c}_1);\mathbf{inj}_? = \mathbf{inj}_\forall(\mathbf{c}_1)$, $\mathbf{inj}_?;\mathbf{inj}_? = \mathbf{inj}_?$ and $\mathbf{inj}_\forall(\mathbf{c}_1);\mathbf{inj}_? = \mathbf{inj}_\forall(\mathbf{c}_1)$, where $\mathbf{c}_1 = \mathbf{c}_4$).
- If $\mathbf{c}_4 = \mathbf{inj}_\rightarrow(\mathbf{c}'_4)$, then by Lemma B.12, we have the following cases.

- $\mathbf{c}_1 = \mathbf{c}_1 \longrightarrow \mathbf{c}_1'', \mathbf{c}_2 = \mathbf{c}_2' \longrightarrow \mathbf{c}_2''$ and $\mathbf{c}_3 = \text{inj}_{\rightarrow}(\mathbf{c}_3' \longrightarrow \mathbf{c}_3'')$. If $(\mathbf{c}_1 \rightarrow \mathbf{c}_1'; \mathbf{c}_2' \rightarrow \mathbf{c}_2''); \text{inj}_{\forall}(\mathbf{c}_3' \longrightarrow \mathbf{c}_3'') = \text{inj}_{\forall}(\mathbf{c}_4' \longrightarrow \mathbf{c}_4'')$, we know by inspecting the rules of gradual type precision transitivity that
 - * there exists \mathbf{c} such that $\mathbf{c}_1 \longrightarrow \mathbf{c}_1'; \mathbf{c}_2' \longrightarrow \mathbf{c}_2'' = \mathbf{c}$, by Definition B.21.
 - * $\mathbf{c} = \mathbf{c}_{11} \longrightarrow \mathbf{c}_{12}$ and $\mathbf{c}_1; \mathbf{c}_2' = \mathbf{c}_{11}$ and $\mathbf{c}_1'; \mathbf{c}_2'' = \mathbf{c}_{12}$.
 - * $\mathbf{c}_{11} \longrightarrow \mathbf{c}_{12}; \text{inj}_{\rightarrow}(\mathbf{c}_3' \longrightarrow \mathbf{c}_3'') = \text{inj}_{\rightarrow}(\mathbf{c}_4' \longrightarrow \mathbf{c}_4'')$, $\mathbf{c}_{11}; \mathbf{c}_3' = \mathbf{c}_4'$ and $\mathbf{c}_{12}; \mathbf{c}_3'' = \mathbf{c}_4''$.
 - * there exists \mathbf{c}'' such that $\mathbf{c}_2' \longrightarrow \mathbf{c}_2''; \text{inj}_{\rightarrow}(\mathbf{c}_3' \longrightarrow \mathbf{c}_3'') = \mathbf{c}''$, by Definition B.21.
 - * $\mathbf{c}'' = \text{inj}_{\rightarrow}(\mathbf{c}_{21} \longrightarrow \mathbf{c}_{22}), \mathbf{c}_2'; \mathbf{c}_3' = \mathbf{c}_{21}$ and $\mathbf{c}_2''; \mathbf{c}_3'' = \mathbf{c}_{22}$.
 - * By the induction hypothesis on \mathbf{c}_4' and \mathbf{c}_4'' , with $(\mathbf{c}_1; \mathbf{c}_2'); \mathbf{c}_3' = \mathbf{c}_4'$ and $(\mathbf{c}_1'; \mathbf{c}_2''); \mathbf{c}_3'' = \mathbf{c}_4''$, we know that $\mathbf{c}_1; (\mathbf{c}_2'; \mathbf{c}_3') = \mathbf{c}_4'$ and $\mathbf{c}_1'; (\mathbf{c}_2''; \mathbf{c}_3'') = \mathbf{c}_4''$.
 - * Thus, we know that $\mathbf{c}_1 \longrightarrow \mathbf{c}_1'; (\mathbf{c}_2' \longrightarrow \mathbf{c}_2''; \text{inj}_{\rightarrow}(\mathbf{c}_3' \longrightarrow \mathbf{c}_3'')) = \text{inj}_{\rightarrow}(\mathbf{c}_4' \longrightarrow \mathbf{c}_4'')$.

In a similar way, if $\mathbf{c}_1 \longrightarrow \mathbf{c}_1'; (\mathbf{c}_2' \longrightarrow \mathbf{c}_2''; \text{inj}_{\rightarrow}(\mathbf{c}_3' \longrightarrow \mathbf{c}_3'')) = \text{inj}_{\rightarrow}(\mathbf{c}_4' \longrightarrow \mathbf{c}_4'')$, we know by inspecting the rules of gradual type precision transitivity that

- * there exists \mathbf{c} such that $\mathbf{c}_2' \longrightarrow \mathbf{c}_2''; \text{inj}_{\rightarrow}(\mathbf{c}_3' \longrightarrow \mathbf{c}_3'') = \mathbf{c}$, by Definition B.21.
- * $\mathbf{c} = \text{inj}_{\rightarrow}(\mathbf{c}_{11} \longrightarrow \mathbf{c}_{12}), \mathbf{c}_2'; \mathbf{c}_3' = \mathbf{c}_{11}$ and $\mathbf{c}_2''; \mathbf{c}_3'' = \mathbf{c}_{12}$.
- * $\mathbf{c}_1 \longrightarrow \mathbf{c}_1'; \text{inj}_{\rightarrow}(\mathbf{c}_{11} \longrightarrow \mathbf{c}_{12}) = \text{inj}_{\rightarrow}(\mathbf{c}_4' \longrightarrow \mathbf{c}_4'')$, $\mathbf{c}_1; \mathbf{c}_{11} = \mathbf{c}_4'$ and $\mathbf{c}_1'; \mathbf{c}_{12} = \mathbf{c}_4''$.
- * there exists \mathbf{c}'' such that $\mathbf{c}_1 \longrightarrow \mathbf{c}_1'; \mathbf{c}_2' \longrightarrow \mathbf{c}_2'' = \mathbf{c}''$, by Definition B.21.
- * $\mathbf{c}'' = \mathbf{c}_{21} \longrightarrow \mathbf{c}_{22}, \mathbf{c}_1; \mathbf{c}_2' = \mathbf{c}_{21}$ and $\mathbf{c}_1'; \mathbf{c}_2'' = \mathbf{c}_{22}$.
- * By the induction hypothesis on \mathbf{c}_4' and \mathbf{c}_4'' , with $\mathbf{c}_1; (\mathbf{c}_2'; \mathbf{c}_3') = \mathbf{c}_4'$ and $\mathbf{c}_1'; (\mathbf{c}_2''; \mathbf{c}_3'') = \mathbf{c}_4''$, we know that $(\mathbf{c}_1; \mathbf{c}_2'); \mathbf{c}_3' = \mathbf{c}_4'$ and $(\mathbf{c}_1'; \mathbf{c}_2''); \mathbf{c}_3'' = \mathbf{c}_4''$.
- * Thus, we know that $(\mathbf{c}_1 \longrightarrow \mathbf{c}_1'; \mathbf{c}_2' \longrightarrow \mathbf{c}_2''); \text{inj}_{\rightarrow}(\mathbf{c}_3' \longrightarrow \mathbf{c}_3'') = \text{inj}_{\rightarrow}(\mathbf{c}_4' \longrightarrow \mathbf{c}_4'')$.

Therefore, the result holds.

- $\mathbf{c}_1 = \mathbf{c}_1 \longrightarrow \mathbf{c}_1'', \mathbf{c}_2 = \text{inj}_{\rightarrow}(\mathbf{c}_2')$ and $\mathbf{c}_3 = \text{inj}_{\rightarrow}$. Straightforward by looking at rules of gradual type precision transitivity ($\mathbf{c}_1; \mathbf{c}_2 = \mathbf{c}_4, \mathbf{c}_2; \text{inj}_{\rightarrow} = \mathbf{c}_2$ and $\mathbf{c}_4; \text{inj}_{\rightarrow} = \mathbf{c}_4$).
- $\mathbf{c}_1 = \text{inj}_{\rightarrow}(\mathbf{c}_1), \mathbf{c}_2 = \text{inj}_{\rightarrow}$ and $\mathbf{c}_3 = \text{inj}_{\rightarrow}$. Straightforward by looking at rules of gradual type precision transitivity ($\mathbf{c}_1; \text{inj}_{\rightarrow} = \mathbf{c}_1, \text{inj}_{\rightarrow}; \text{inj}_{\rightarrow} = \text{inj}_{\rightarrow}$ and $\mathbf{c}_1; \text{inj}_{\rightarrow} = \mathbf{c}_1$, where $\mathbf{c}_1 = \mathbf{c}_4$).
- If $\mathbf{c}_4 = \text{inj}_{\rightarrow}$, then by inspecting the rules of gradual type precision transitivity, we know that $\mathbf{c}_1 = \text{inj}_{\rightarrow}, \mathbf{c}_2 = \text{inj}_{\rightarrow}$ and $\mathbf{c}_3 = \text{inj}_{\rightarrow}$, and the result follows straightforwardly.

□

Lemma B.14 *If*

- $\mathbf{c} : \mathbf{G} \sqsubseteq \mathbf{G}$
- $\mathbf{c}' : \mathbf{G} \sqsubseteq \mathbf{G}'$

then for all \mathbf{c}'' such that $\mathbf{c}; \mathbf{c}' = \mathbf{c}''$, we get that $\mathbf{c}'' = \mathbf{c}'$.

PROOF. We proceed by induction on $\mathbf{c} : \mathbf{G} \sqsubseteq \mathbf{G}$. We have the following cases by inspecting the syntax of $\mathbf{c} : \mathbf{G} \sqsubseteq \mathbf{G}$.

- If $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$, where $\mathbf{c} = \mathbf{B}$ and $\mathbf{G} = \mathbf{B}$, then we have the following cases by inspecting the type precision transitivity rules.
 - If $\mathbf{c}' = \mathbf{inj}_B$, $\mathbf{G}' = ?_\delta$ and $\mathbf{inj}_B : \mathbf{B} \sqsubseteq ?_\delta$, then we know that the only type precision transitivity rule applicable is $\mathbf{B};\mathbf{inj}_B = \mathbf{inj}_B$, where $\mathbf{c}'' = \mathbf{c}' = \mathbf{inj}_B$. Thus, the result follows.
 - If $\mathbf{c}' = \mathbf{inj}_X$, $\mathbf{G}' = ?_\delta$, $\mathbf{inj}_X : \mathbf{B} \sqsubseteq ?_\delta$ and $\mathbf{X} : \mathbf{B} \in \delta$, then we know that the only type precision transitivity rule applicable is $\mathbf{B};\mathbf{inj}_X = \mathbf{inj}_X$, where $\mathbf{c}'' = \mathbf{c}' = \mathbf{inj}_X$. Thus, the result follows.
- If $\mathbf{inj}_? : ?_\delta \sqsubseteq ?_\delta$, where $\mathbf{c} = \mathbf{inj}_?$ and $\mathbf{G} = ?_\delta$, then we have the following cases by inspecting the type precision transitivity rules.
 - If $\mathbf{c}' = \mathbf{inj}_?$, $\mathbf{G}' = ?_{\delta'}$, $\mathbf{inj}_? : ?_\delta \sqsubseteq ?_{\delta'}$ and $\delta \subseteq \delta'$, then we know that the only type precision transitivity rule applicable is $\mathbf{inj}_?;\mathbf{inj}_? = \mathbf{inj}_?$, where $\mathbf{c}'' = \mathbf{c}' = \mathbf{inj}_?$. Thus, the result follows. Thus, the result follows.
- If $\forall \mathbf{X}.\mathbf{c}_1 : \forall \mathbf{X}.\mathbf{G}_1 \sqsubseteq \forall \mathbf{X}.\mathbf{G}_1$, where $\mathbf{c} = \forall \mathbf{X}.\mathbf{c}_1$, $\mathbf{G} = \forall \mathbf{X}.\mathbf{G}_1$ and $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_1$, then we have the following cases by inspecting the type precision transitivity rules.
 - If $\mathbf{c}' = \forall \mathbf{X}.\mathbf{c}_1$ and $\mathbf{G}' = \forall \mathbf{X}.\mathbf{G}'_1$, where $\forall \mathbf{X}.\mathbf{c}_1 : \forall \mathbf{X}.\mathbf{G}_1 \sqsubseteq \forall \mathbf{X}.\mathbf{G}'_1$ and $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}'_1$, then we apply the induction hypothesis on $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_1$ with $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}'_1$, and we get that for all \mathbf{c}''_1 such that $\mathbf{c}_1;\mathbf{c}_1 = \mathbf{c}''_1$, $\mathbf{c}_1 = \mathbf{c}''_1$. Therefore, we get that for all $\forall \mathbf{X}.\mathbf{c}''_1$ such that $\forall \mathbf{X}.\mathbf{c}_1;\forall \mathbf{X}.\mathbf{c}_1 = \forall \mathbf{X}.\mathbf{c}''_1$, $\forall \mathbf{X}.\mathbf{c}_1 = \forall \mathbf{X}.\mathbf{c}''_1$. Thus, the result follows.
 - If $\mathbf{c}' = \mathbf{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_1)$ and $\mathbf{G}' = ?_\delta$, where $\mathbf{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_1) : \forall \mathbf{X}.\mathbf{G}_1 \sqsubseteq ?_\delta$, $\forall \mathbf{X}.\mathbf{c}_1 : \forall \mathbf{X}.\mathbf{G}_1 \sqsubseteq \forall \mathbf{X}?._\delta.X:X$ and $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq ?_{\delta,X:X}$, then we apply the induction hypothesis on $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_1$ with $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq ?_{\delta,X:X}$, and we get that for all \mathbf{c}''_1 such that $\mathbf{c}_1;\mathbf{c}_1 = \mathbf{c}''_1$, $\mathbf{c}_1 = \mathbf{c}''_1$. Therefore, we get that for all $\forall \mathbf{X}.\mathbf{c}''_1$ such that $\forall \mathbf{X}.\mathbf{c}_1;\forall \mathbf{X}.\mathbf{c}_1 = \forall \mathbf{X}.\mathbf{c}''_1$, $\forall \mathbf{X}.\mathbf{c}_1 = \forall \mathbf{X}.\mathbf{c}''_1$. Hence, we get that for all $\mathbf{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}''_1)$ such that $\forall \mathbf{X}.\mathbf{c}_1;\mathbf{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_1) = \mathbf{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}''_1)$, $\mathbf{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_1) = \mathbf{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}''_1)$. Thus, the result follows.
- If $\mathbf{c}_1 \rightarrow \mathbf{c}_2 : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \sqsubseteq \mathbf{G}_1 \rightarrow \mathbf{G}_2$, where $\mathbf{c} = \mathbf{c}_1 \rightarrow \mathbf{c}_2$, $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$, $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_1$ and $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}_2$, then we have the following cases by inspecting the type precision transitivity rules.
 - If $\mathbf{c}' = \mathbf{c}_1 \rightarrow \mathbf{c}'_2$ and $\mathbf{G}' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$, where $\mathbf{c}_1 \rightarrow \mathbf{c}'_2 : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \sqsubseteq \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$, $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}'_1$ and $\mathbf{c}'_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}'_2$, then we apply the induction hypothesis on $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_1$ and $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}_2$ with $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}'_1$ and $\mathbf{c}'_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}'_2$, and we get that for all \mathbf{c}''_1 and \mathbf{c}''_2 such that $\mathbf{c}_1;\mathbf{c}_1 = \mathbf{c}''_1$ and $\mathbf{c}_2;\mathbf{c}'_2 = \mathbf{c}''_2$, $\mathbf{c}_1 = \mathbf{c}''_1$ and $\mathbf{c}'_2 = \mathbf{c}''_2$. Therefore, we get that for all $\mathbf{c}''_1 \rightarrow \mathbf{c}''_2$ such that $\mathbf{c}_1 \rightarrow \mathbf{c}_2;\mathbf{c}_1 \rightarrow \mathbf{c}'_2 = \mathbf{c}''_1 \rightarrow \mathbf{c}''_2$, $\mathbf{c}_1 \rightarrow \mathbf{c}'_2 = \mathbf{c}''_1 \rightarrow \mathbf{c}''_2$. Thus, the result follows.
 - If $\mathbf{c}' = \mathbf{inj}_{\rightarrow}(\mathbf{c}_1 \rightarrow \mathbf{c}'_2)$ and $\mathbf{G}' = ?_\delta$, where $\mathbf{inj}_{\rightarrow}(\mathbf{c}_1 \rightarrow \mathbf{c}'_2) : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \sqsubseteq ?_\delta$, $\mathbf{c}_1 \rightarrow \mathbf{c}'_2 : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \sqsubseteq ?_\delta \rightarrow ?_\delta$, $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq ?_\delta$ and $\mathbf{c}'_2 : \mathbf{G}_2 \sqsubseteq ?_\delta$, then we apply the induction hypothesis on $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_1$ and $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}_2$ with $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq ?_\delta$ and $\mathbf{c}'_2 : \mathbf{G}_2 \sqsubseteq ?_\delta$, and we get that for all \mathbf{c}''_1 and \mathbf{c}''_2 such that $\mathbf{c}_1;\mathbf{c}_1 = \mathbf{c}''_1$ and $\mathbf{c}_2;\mathbf{c}'_2 = \mathbf{c}''_2$, $\mathbf{c}_1 = \mathbf{c}''_1$ and $\mathbf{c}'_2 = \mathbf{c}''_2$. Therefore, we get that for all $\mathbf{c}''_1 \rightarrow \mathbf{c}''_2$ such that $\mathbf{c}_1 \rightarrow \mathbf{c}_2;\mathbf{c}_1 \rightarrow \mathbf{c}'_2 = \mathbf{c}''_1 \rightarrow \mathbf{c}''_2$, $\mathbf{c}_1 \rightarrow \mathbf{c}'_2 = \mathbf{c}''_1 \rightarrow \mathbf{c}''_2$. Hence, we get that for all $\mathbf{inj}_{\rightarrow}(\mathbf{c}''_1 \rightarrow \mathbf{c}''_2)$ such that $\mathbf{c}_1 \rightarrow \mathbf{c}_2;\mathbf{inj}_{\rightarrow}(\mathbf{c}_1 \rightarrow \mathbf{c}'_2) = \mathbf{inj}_{\rightarrow}(\mathbf{c}''_1 \rightarrow \mathbf{c}''_2)$, $\mathbf{inj}_{\rightarrow}(\mathbf{c}_1 \rightarrow \mathbf{c}'_2) = \mathbf{inj}_{\rightarrow}(\mathbf{c}''_1 \rightarrow \mathbf{c}''_2)$. Thus, the result follows.

□

Lemma B.15 *If $\mathbf{c} : \mathbf{G} \sqsubseteq ?_{\delta}$ and $\delta \subseteq \delta'$ then $\mathbf{c} : \mathbf{G} \sqsubseteq ?_{\delta'}$.*

PROOF. By induction on syntax of \mathbf{c} . We have the following cases such that $\mathbf{c} : \mathbf{G} \sqsubseteq ?_{\delta}$.

- If $\mathbf{c} = \text{inj}_{\mathbf{X}}$, where $\text{inj}_{\mathbf{X}} : \mathbf{F} \sqsubseteq ?_{\delta}$ and $\mathbf{X} : \mathbf{F} \in \delta$, then we know that $\text{inj}_{\mathbf{X}} : \mathbf{F} \sqsubseteq ?_{\delta'}$, since $\delta \subseteq \delta'$, we get that $\mathbf{X} : \mathbf{F} \in \delta'$. Thus, the result holds.
- If $\mathbf{c} = \text{inj}_{\mathbf{B}}$, where $\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta}$, then the result holds because $\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta'}$, for any $?_{\delta'}$.
- If $\mathbf{c} = \text{inj}_{?}$, where $\text{inj}_{?} : ?_{\delta''} \sqsubseteq ?_{\delta}$, then the result holds since $\delta'' \subseteq \delta$ and $\delta \subseteq \delta'$, we get that $\delta'' \subseteq \delta'$.
- If $\mathbf{c} = \text{inj}_{\rightarrow}(\mathbf{c})$, where $\text{inj}_{\rightarrow}(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_{\delta}$, $\mathbf{c}' : \mathbf{G} \sqsubseteq ?_{\delta} \rightarrow ?_{\delta}$, then we know that $\mathbf{G} = \mathbf{G}' \rightarrow \mathbf{G}''$, $\mathbf{c}' = \mathbf{c}'' \rightarrow \mathbf{c}'''$, $\mathbf{c}'' : \mathbf{G}' \sqsubseteq ?_{\delta}$ and $\mathbf{c}''' : \mathbf{G}'' \sqsubseteq ?_{\delta}$. Thus, by the induction hypothesis on \mathbf{c}'' and \mathbf{c}''' , with $\mathbf{c}'' : \mathbf{G}' \sqsubseteq ?_{\delta}$ and $?_{\delta'}$, and $\mathbf{c}''' : \mathbf{G}'' \sqsubseteq ?_{\delta}$ and $?_{\delta'}$, we get that $\mathbf{c}'' : \mathbf{G}' \sqsubseteq ?_{\delta'}$ and $\mathbf{c}''' : \mathbf{G}'' \sqsubseteq ?_{\delta'}$, therefore $\mathbf{c}'' \rightarrow \mathbf{c}''' : \mathbf{G}' \rightarrow \mathbf{G}'' \sqsubseteq ?_{\delta'} \rightarrow ?_{\delta'}$ and $\text{inj}_{\rightarrow}(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_{\delta'}$. Thus, the result follows.
- If $\mathbf{c} = \text{inj}_{\forall}(\mathbf{c}')$, where $\text{inj}_{\forall}(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_{\delta}$, $\mathbf{c}' : \mathbf{G} \sqsubseteq \forall \mathbf{X}. ?_{\delta, \mathbf{X} : \mathbf{X}}$, then we know that $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}'$, $\mathbf{c}' = \forall \mathbf{X}. \mathbf{c}''$ and $\mathbf{c}'' : \mathbf{G}' \sqsubseteq ?_{\delta, \mathbf{X} : \mathbf{X}}$. Thus, by the induction hypothesis on \mathbf{c}'' , with $\mathbf{c}'' : \mathbf{G}' \sqsubseteq ?_{\delta, \mathbf{X} : \mathbf{X}}$ and $?_{\delta', \mathbf{X} : \mathbf{X}}$, we get that $\mathbf{c}'' : \mathbf{G}' \sqsubseteq ?_{\delta', \mathbf{X} : \mathbf{X}}$, therefore $\forall \mathbf{X}. \mathbf{c}'' : \forall \mathbf{X}. \mathbf{G}' \sqsubseteq \forall \mathbf{X}. ?_{\delta', \mathbf{X} : \mathbf{X}}$ and $\text{inj}_{\forall}(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_{\delta'}$. Thus, the result follows.

□

B.2.2 Pullback Operator

Definition B.16 ((Partial) Precision meet operator) *If*

- $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}$
- $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}$

Then we will define

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) \in \{(\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \mid \mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1, \mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2, \mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5 \text{ and } \mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}_5\}^*$$

We define the function by induction on the precision judgements $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}$ and $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}$:

$$\begin{aligned}
& \text{pullback}(\mathbf{B}, (\mathbf{B}, \mathbf{B}), (\mathbf{B}, \mathbf{B})) = (\mathbf{B}, \mathbf{B}, \mathbf{B}, \mathbf{B}) \\
& \text{pullback}(\mathbf{X}, (\mathbf{X}, \mathbf{X}), (\mathbf{X}, \mathbf{X})) = (\mathbf{X}, \mathbf{X}, \mathbf{X}, \mathbf{X}) \\
& \text{pullback}(\forall \mathbf{X}.\mathbf{G}, (\forall \mathbf{X}.\mathbf{G}_1, \forall \mathbf{X}.\mathbf{c}_1), (\forall \mathbf{X}.\mathbf{G}_2, \forall \mathbf{X}.\mathbf{c}_2)) = \begin{cases} (\forall \mathbf{X}.\mathbf{G}_3, \forall \mathbf{X}.\mathbf{c}_3, \forall \mathbf{X}.\mathbf{c}_4, \forall \mathbf{X}.\mathbf{c}_5) \text{ if} \\ \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \end{cases} \\
& \text{pullback} \left(\begin{array}{l} \mathbf{G} \longrightarrow \mathbf{G}', (\mathbf{G}_1 \rightarrow \mathbf{G}'_1, \mathbf{c}_1 \rightarrow \mathbf{c}'_1), \\ (\mathbf{G}_2 \longrightarrow \mathbf{G}'_2, \mathbf{c}_2 \longrightarrow \mathbf{c}'_2) \end{array} \right) = \begin{cases} (\mathbf{G}_3 \longrightarrow \mathbf{G}'_3, \mathbf{c}_3 \longrightarrow \mathbf{c}'_3, \mathbf{c}_4 \longrightarrow \mathbf{c}'_4, \mathbf{c}_5 \longrightarrow \mathbf{c}'_5) \text{ if} \\ \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \wedge \\ \text{pullback}(\mathbf{G}', (\mathbf{G}'_1, \mathbf{c}'_1), (\mathbf{G}'_2, \mathbf{c}'_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5) \end{cases} \\
& \text{pullback} \left(\begin{array}{l} \mathbf{G} \times \mathbf{G}', (\mathbf{G}_1 \times \mathbf{G}'_1, \mathbf{c}_1 \times \mathbf{c}'_1), \\ (\mathbf{G}_2 \times \mathbf{G}'_2, \mathbf{c}_2 \times \mathbf{c}'_2) \end{array} \right) = \begin{cases} (\mathbf{G}_3 \times \mathbf{G}'_3, \mathbf{c}_3 \times \mathbf{c}'_3, \mathbf{c}_4 \times \mathbf{c}'_4, \mathbf{c}_5 \times \mathbf{c}'_5) \text{ if} \\ \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \wedge \\ \text{pullback}(\mathbf{G}', (\mathbf{G}'_1, \mathbf{c}'_1), (\mathbf{G}'_2, \mathbf{c}'_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5) \end{cases} \\
& \text{pullback}(\delta, (\mathbf{B}, \text{inj}_{\mathbf{B}}), (\mathbf{B}, \text{inj}_{\mathbf{B}})) = (\mathbf{B}, \mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{B}}) \\
& \text{pullback}(\delta, (\mathbf{F}, \text{inj}_{\mathbf{X}}), (\mathbf{F}, \text{inj}_{\mathbf{X}})) = (\mathbf{F}, \mathbf{F}, \mathbf{F}, \text{inj}_{\mathbf{X}}) \text{ if } \mathbf{X} : \mathbf{F} \in \delta \\
& \text{pullback}(\delta, (\mathbf{G}_1, \text{inj}_{\vee}(\mathbf{c}_1)), (\mathbf{G}_2, \text{inj}_{\vee}(\mathbf{c}_2))) = \begin{cases} (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \text{inj}_{\vee}(\mathbf{c}_5)) \text{ if} \\ \text{pullback}(\forall \mathbf{X}.\delta, \mathbf{X}:\mathbf{X}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \end{cases} \\
& \text{pullback}(\delta, (\mathbf{G}_1, \text{inj}_{\rightarrow}(\mathbf{c}_1)), (\mathbf{G}_2, \text{inj}_{\rightarrow}(\mathbf{c}_2))) = \begin{cases} (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \text{inj}_{\rightarrow}(\mathbf{c}_5)) \text{ if} \\ \text{pullback}(\delta \rightarrow \delta, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \end{cases} \\
& \text{pullback}(\delta, (\mathbf{G}_1, \text{inj}_{\times}(\mathbf{c}_1)), (\mathbf{G}_2, \text{inj}_{\times}(\mathbf{c}_2))) = \begin{cases} (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \text{inj}_{\times}(\mathbf{c}_5)) \text{ if} \\ \text{pullback}(\delta \times \delta, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \end{cases} \\
& \text{pullback}(\delta, (\delta', \text{inj}_{\delta'}), (\mathbf{G}_2, \mathbf{c}_2)) = \begin{cases} (\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B}, \text{inj}_{\mathbf{B}}) \text{ if } \mathbf{G}_2 = \mathbf{B} \wedge \mathbf{c}_2 = \text{inj}_{\mathbf{B}} \\ (\mathbf{F}, \text{inj}_{\mathbf{X}}, \mathbf{F}, \text{inj}_{\mathbf{X}}) \text{ if } \mathbf{G}_2 = \mathbf{F} \wedge \mathbf{c}_2 = \text{inj}_{\mathbf{X}} \wedge \mathbf{X} : \mathbf{F} \in \delta' \\ (\mathbf{G}_3, \text{inj}_{\vee}(\mathbf{c}_3), \mathbf{c}_4, \text{inj}_{\vee}(\mathbf{c}_5)) \text{ if } \mathbf{c}_2 = \text{inj}_{\vee}(\mathbf{c}) \\ \text{pullback}(\forall \mathbf{X}.\delta, \mathbf{X}:\mathbf{X}, (\forall \mathbf{X}.\delta', \mathbf{X}:\mathbf{X}, \forall \mathbf{X}.\text{inj}_{\delta'}), (\mathbf{G}_2, \mathbf{c})) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \\ (\mathbf{G}_3, \text{inj}_{\rightarrow}(\mathbf{c}_3), \mathbf{c}_4, \text{inj}_{\rightarrow}(\mathbf{c}_5)) \text{ if } \mathbf{c}_2 = \text{inj}_{\rightarrow}(\mathbf{c}) \\ \text{pullback}(\delta \rightarrow \delta, (\delta' \rightarrow \delta', \text{inj}_{\delta'} \rightarrow \text{inj}_{\delta'}), (\mathbf{G}_2, \mathbf{c})) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \\ (\mathbf{G}_3, \text{inj}_{\times}(\mathbf{c}_3), \mathbf{c}_4, \text{inj}_{\times}(\mathbf{c}_5)) \text{ if } \mathbf{c}_2 = \text{inj}_{\times}(\mathbf{c}) \\ \text{pullback}(\delta \times \delta, (\delta' \times \delta', \text{inj}_{\delta'} \times \text{inj}_{\delta'}), (\mathbf{G}_2, \mathbf{c})) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \\ (\delta' \cap \delta'', \text{inj}_{\delta'}, \text{inj}_{\delta'}, \text{inj}_{\delta'}) \text{ if } \mathbf{G}_2 = \delta'' \wedge \mathbf{c}_2 = \text{inj}_{\delta'} \end{cases} \\
& \text{pullback}(\delta, (\mathbf{G}_1, \mathbf{c}_1), (\delta', \text{inj}_{\delta'})) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \text{ if } \text{pullback}(\delta, (\delta', \text{inj}_{\delta'}), (\mathbf{G}_1, \mathbf{c}_1)) = (\mathbf{G}_3, \mathbf{c}_4, \mathbf{c}_3, \mathbf{c}_5) \\
& \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{undefined otherwise}
\end{aligned}$$

Lemma B.17 (Pullback operator and its universal property) *There exists a partial pullback operator such that if $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}$ and $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}$, and $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$, then $\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1$, $\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2$, $\mathbf{c}_3;\mathbf{c}_1 = \mathbf{c}_5$ and $\mathbf{c}_4;\mathbf{c}_2 = \mathbf{c}_5$. The pullback operator is universal in the following sense. If there exists $\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4$ and \mathbf{c}'_5 such that $\mathbf{c}'_3;\mathbf{c}_1 = \mathbf{c}'_5$ and $\mathbf{c}'_4;\mathbf{c}_2 = \mathbf{c}'_5$, then $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$ and there exists a unique $\mathbf{c} : \mathbf{G}_3 \sqsubseteq \mathbf{G}_3$ such that $\mathbf{c};\mathbf{c}_3 = \mathbf{c}'_3$, $\mathbf{c};\mathbf{c}_4 = \mathbf{c}'_4$ and $\mathbf{c};\mathbf{c}_5 = \mathbf{c}'_5$.*

PROOF. We proceed by induction on $\mathbf{c}'_5 : \mathbf{G}'_3 \sqsubseteq \mathbf{G}$ and by Lemma B.12 with $\mathbf{c}'_3; \mathbf{c}_1 = \mathbf{c}'_5$ and $\mathbf{c}'_4; \mathbf{c}_2 = \mathbf{c}'_5$.

1. $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$ ($\mathbf{c}'_5 : \mathbf{G}'_3 \sqsubseteq \mathbf{G}$). Then, $\mathbf{c}'_5 = \mathbf{B}$ and $\mathbf{G}'_3 = \mathbf{G} = \mathbf{B}$. Since $\mathbf{c}'_3; \mathbf{c}_1 = \mathbf{c}'_5$ and $\mathbf{c}'_4; \mathbf{c}_2 = \mathbf{c}'_5$, by Lemma B.12, we know that $\mathbf{c}'_3 = \mathbf{c}_1 = \mathbf{c}'_4 = \mathbf{c}_2 = \mathbf{B}$ and $\mathbf{G}_1 = \mathbf{G}_2 = \mathbf{B}$. Therefore, we know that $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\mathbf{B}, (\mathbf{B}, \mathbf{B}), (\mathbf{B}, \mathbf{B})) = (\mathbf{B}, \mathbf{B}, \mathbf{B}, \mathbf{B})$. Thus, we know that $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{B}, \mathbf{B}, \mathbf{B}, \mathbf{B})$ and there exist a unique $\mathbf{c} = \mathbf{B}$ such that $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$ ($\mathbf{c} : \mathbf{G}'_3 \sqsubseteq \mathbf{G}_3$) and

- $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$ ($\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1$)
- $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$ ($\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2$)
- $\mathbf{B}; \mathbf{B} = \mathbf{B}$ ($\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5$)
- $\mathbf{B}; \mathbf{B} = \mathbf{B}$ ($\mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}_5$)
- $\mathbf{B}; \mathbf{B} = \mathbf{B}$ ($\mathbf{c}; \mathbf{c}_3 = \mathbf{c}'_3$)
- $\mathbf{B}; \mathbf{B} = \mathbf{B}$ ($\mathbf{c}; \mathbf{c}_4 = \mathbf{c}'_4$)
- $\mathbf{B}; \mathbf{B} = \mathbf{B}$ ($\mathbf{c}; \mathbf{c}_5 = \mathbf{c}'_5$)

2. $\mathbf{X} : \mathbf{X} \sqsubseteq \mathbf{X}$ ($\mathbf{c}'_5 : \mathbf{G}'_3 \sqsubseteq \mathbf{G}$). Then, $\mathbf{c}'_5 = \mathbf{X}$ and $\mathbf{G}'_3 = \mathbf{G} = \mathbf{X}$. Since $\mathbf{c}'_3; \mathbf{c}_1 = \mathbf{c}'_5$ and $\mathbf{c}'_4; \mathbf{c}_2 = \mathbf{c}'_5$, by Lemma B.12, we know that $\mathbf{c}'_3 = \mathbf{c}_1 = \mathbf{c}'_4 = \mathbf{c}_2 = \mathbf{X}$ and $\mathbf{G}_1 = \mathbf{G}_2 = \mathbf{X}$. Therefore, we know that $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\mathbf{X}, (\mathbf{X}, \mathbf{X}), (\mathbf{X}, \mathbf{X})) = (\mathbf{X}, \mathbf{X}, \mathbf{X}, \mathbf{X})$. Thus, we know that $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{X}, \mathbf{X}, \mathbf{X}, \mathbf{X})$ and there exist a unique $\mathbf{c} = \mathbf{X}$ such that $\mathbf{X} : \mathbf{X} \sqsubseteq \mathbf{X}$ ($\mathbf{c} : \mathbf{G}'_3 \sqsubseteq \mathbf{G}_3$) and

- $\mathbf{X} : \mathbf{X} \sqsubseteq \mathbf{X}$ ($\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1$)
- $\mathbf{X} : \mathbf{X} \sqsubseteq \mathbf{X}$ ($\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2$)
- $\mathbf{X}; \mathbf{X} = \mathbf{X}$ ($\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5$)
- $\mathbf{X}; \mathbf{X} = \mathbf{X}$ ($\mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}_5$)
- $\mathbf{X}; \mathbf{X} = \mathbf{X}$ ($\mathbf{c}; \mathbf{c}_3 = \mathbf{c}'_3$)
- $\mathbf{X}; \mathbf{X} = \mathbf{X}$ ($\mathbf{c}; \mathbf{c}_4 = \mathbf{c}'_4$)
- $\mathbf{X}; \mathbf{X} = \mathbf{X}$ ($\mathbf{c}; \mathbf{c}_5 = \mathbf{c}'_5$)

3. $\forall \mathbf{X}. \mathbf{c}'_{a5} : \forall \mathbf{X}. \mathbf{G}'_{3a} \sqsubseteq \forall \mathbf{X}. \mathbf{G}_a$ ($\mathbf{c}'_5 : \mathbf{G}'_3 \sqsubseteq \mathbf{G}$). Then, $\mathbf{c}'_5 = \forall \mathbf{X}. \mathbf{c}'_{a5}$, $\mathbf{G}'_3 = \forall \mathbf{X}. \mathbf{G}'_{3a}$ and $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}_a$. Since $\mathbf{c}'_3; \mathbf{c}_1 = \mathbf{c}'_5$ and $\mathbf{c}'_4; \mathbf{c}_2 = \mathbf{c}'_5$, by Lemma B.12, we know that $\mathbf{c}'_3 = \forall \mathbf{X}. \mathbf{c}'_{3a}$, $\mathbf{c}_1 = \forall \mathbf{X}. \mathbf{c}_{a1}$, $\mathbf{c}'_4 = \forall \mathbf{X}. \mathbf{c}'_{a4}$, $\mathbf{c}_2 = \forall \mathbf{X}. \mathbf{c}_{a2}$, $\mathbf{G}_1 = \forall \mathbf{X}. \mathbf{G}_{a1}$ and $\mathbf{G}_2 = \forall \mathbf{X}. \mathbf{G}_{a2}$, where

- $\mathbf{c}_{a1} : \mathbf{G}_{a1} \sqsubseteq \mathbf{G}_a$
- $\mathbf{c}_{a2} : \mathbf{G}_{a2} \sqsubseteq \mathbf{G}_a$
- $\mathbf{c}'_{a3} : \mathbf{G}'_{a3} \sqsubseteq \mathbf{G}_{a1}$
- $\mathbf{c}'_{a4} : \mathbf{G}'_{a3} \sqsubseteq \mathbf{G}_{a2}$
- $\mathbf{c}'_{a5} : \mathbf{G}'_{a3} \sqsubseteq \mathbf{G}_a$
- $\mathbf{c}'_{a3}; \mathbf{c}_{a1} = \mathbf{c}'_{a5}$
- $\mathbf{c}'_{a4}; \mathbf{c}_{a2} = \mathbf{c}'_{a5}$

By the induction hypothesis on $\mathbf{c}'_{a5} : \mathbf{G}'_{a3} \sqsubseteq \mathbf{G}_a$, we know that $\text{pullback}(\mathbf{G}_a, (\mathbf{G}_{a1}, \mathbf{c}_{a1}), (\mathbf{G}_{a2}, \mathbf{c}_{a2})) = (\mathbf{G}_{a3}, \mathbf{c}_{a3}, \mathbf{c}_{a4}, \mathbf{c}_{a5})$ and there exists a unique $\mathbf{c}_a : \mathbf{G}'_{a3} \sqsubseteq \mathbf{G}_{a3}$ such that

- $\mathbf{c}_{a3} : \mathbf{G}_{a3} \sqsubseteq \mathbf{G}_{a1}$
- $\mathbf{c}_{a4} : \mathbf{G}_{a3} \sqsubseteq \mathbf{G}_{a2}$
- $\mathbf{c}_{a3};\mathbf{c}_{a1} = \mathbf{c}_{a5}$
- $\mathbf{c}_{a4};\mathbf{c}_{a2} = \mathbf{c}_{a5}$
- $\mathbf{c}_a;\mathbf{c}_{a3} = \mathbf{c}'_{a3}$
- $\mathbf{c}_a;\mathbf{c}_{a4} = \mathbf{c}'_{a4}$
- $\mathbf{c}_a;\mathbf{c}_{a5} = \mathbf{c}'_{a5}$

We know, by the definition of meet B.16, that

$$\begin{aligned} & \text{pullback}(\forall \mathbf{X}.\mathbf{G}_a, (\forall \mathbf{X}.\mathbf{G}_{a1}, \forall \mathbf{X}.\mathbf{c}_{a1}), (\forall \mathbf{X}.\mathbf{G}_{a2}, \forall \mathbf{X}.\mathbf{c}_{a2})) = \\ & (\forall \mathbf{X}.\mathbf{G}_{c3}, \forall \mathbf{X}.\mathbf{c}_{c3}, \forall \mathbf{X}.\mathbf{c}_{c4}, \forall \mathbf{X}.\mathbf{c}_{c5}) \mid (\mathbf{G}_{c3}, \mathbf{c}_{c3}, \mathbf{c}_{c4}, \mathbf{c}_{c5}) = \text{pullback}(\mathbf{G}_a, (\mathbf{G}_{a1}, \mathbf{c}_{a1}), (\mathbf{G}_{a2}, \mathbf{c}_{a2})) \end{aligned}$$

Therefore, by the definition of meet B.16, we know that

$$\begin{aligned} & (\forall \mathbf{X}.\mathbf{G}_{a3}, \forall \mathbf{X}.\mathbf{c}_{a3}, \forall \mathbf{X}.\mathbf{c}_{a4}, \forall \mathbf{X}.\mathbf{c}_{a5}) \\ & = \text{pullback}(\forall \mathbf{X}.\mathbf{G}_a, (\forall \mathbf{X}.\mathbf{G}_{a1}, \forall \mathbf{X}.\mathbf{c}_{a1}), (\forall \mathbf{X}.\mathbf{G}_{a2}, \forall \mathbf{X}.\mathbf{c}_{a2})) \end{aligned}$$

and there exists a unique $\forall \mathbf{X}.\mathbf{c}_a : \forall \mathbf{X}.\mathbf{G}'_{a3} \sqsubseteq \forall \mathbf{X}.\mathbf{G}_{a3}$ such that

- $\forall \mathbf{X}.\mathbf{c}_{a3} : \forall \mathbf{X}.\mathbf{G}_{a3} \sqsubseteq \forall \mathbf{X}.\mathbf{G}_{a1}$
- $\forall \mathbf{X}.\mathbf{c}_{a4} : \forall \mathbf{X}.\mathbf{G}_{a3} \sqsubseteq \forall \mathbf{X}.\mathbf{G}_{a2}$
- $\forall \mathbf{X}.\mathbf{c}_{a3};\forall \mathbf{X}.\mathbf{c}_{a1} = \forall \mathbf{X}.\mathbf{c}_{a5}$
- $\forall \mathbf{X}.\mathbf{c}_{a4};\forall \mathbf{X}.\mathbf{c}_{a2} = \forall \mathbf{X}.\mathbf{c}_{a5}$
- $\forall \mathbf{X}.\mathbf{c}_a;\forall \mathbf{X}.\mathbf{c}_{a3} = \forall \mathbf{X}.\mathbf{c}'_{a3}$
- $\forall \mathbf{X}.\mathbf{c}_a;\forall \mathbf{X}.\mathbf{c}_{a4} = \forall \mathbf{X}.\mathbf{c}'_{a4}$
- $\forall \mathbf{X}.\mathbf{c}_a;\forall \mathbf{X}.\mathbf{c}_{a5} = \forall \mathbf{X}.\mathbf{c}'_{a5}$

Thus, the result holds.

4. $\mathbf{c}'_{a5} \longrightarrow \mathbf{c}'_{b5} : \mathbf{G}'_{a3} \longrightarrow \mathbf{G}'_{b3} \sqsubseteq \mathbf{G}_a \longrightarrow \mathbf{G}_b$ ($\mathbf{c}'_5 : \mathbf{G}'_3 \sqsubseteq \mathbf{G}$). Then, $\mathbf{c}'_5 = \mathbf{c}'_{a5} \longrightarrow \mathbf{c}'_{b5}$, $\mathbf{G}'_3 = \mathbf{G}'_{3a} \longrightarrow \mathbf{G}'_{3b}$ and $\mathbf{G} = \mathbf{G}_a \longrightarrow \mathbf{G}_b$. Since $\mathbf{c}'_3;\mathbf{c}_1 = \mathbf{c}'_5$ and $\mathbf{c}'_4;\mathbf{c}_2 = \mathbf{c}'_5$, by Lemma B.12, we know that $\mathbf{c}'_3 = \mathbf{c}'_{3a} \longrightarrow \mathbf{c}'_{3b}$, $\mathbf{c}_1 = \mathbf{c}_{a1} \longrightarrow \mathbf{c}_{b1}$, $\mathbf{c}'_4 = \mathbf{c}'_{a4} \longrightarrow \mathbf{c}'_{b4}$, $\mathbf{c}_2 = \mathbf{c}_{a2} \longrightarrow \mathbf{c}_{b2}$, $\mathbf{G}_1 = \mathbf{G}_{a1} \longrightarrow \mathbf{G}_{b1}$ and $\mathbf{G}_2 = \mathbf{G}_{a2} \longrightarrow \mathbf{G}_{b2}$, where

- $\mathbf{c}_{a1} : \mathbf{G}_{a1} \sqsubseteq \mathbf{G}_a$
- $\mathbf{c}_{a2} : \mathbf{G}_{a2} \sqsubseteq \mathbf{G}_a$
- $\mathbf{c}'_{a3} : \mathbf{G}'_{a3} \sqsubseteq \mathbf{G}_{a1}$
- $\mathbf{c}'_{a4} : \mathbf{G}'_{a3} \sqsubseteq \mathbf{G}_{a2}$
- $\mathbf{c}'_{a5} : \mathbf{G}'_{a3} \sqsubseteq \mathbf{G}_a$
- $\mathbf{c}'_{a3};\mathbf{c}_{a1} = \mathbf{c}'_{a5}$
- $\mathbf{c}'_{a4};\mathbf{c}_{a2} = \mathbf{c}'_{a5}$
- $\mathbf{c}_{b1} : \mathbf{G}_{b1} \sqsubseteq \mathbf{G}_b$
- $\mathbf{c}_{b2} : \mathbf{G}_{b2} \sqsubseteq \mathbf{G}_b$

- $c'_{b3} : G'_{b3} \sqsubseteq G_{b1}$
- $c'_{b4} : G'_{b3} \sqsubseteq G_{b2}$
- $c'_{b5} : G'_{b3} \sqsubseteq G_a$
- $c'_{b3};c_{b1} = c'_{b5}$
- $c'_{b4};c_{b2} = c'_{b5}$

By the induction hypothesis on $c'_{a5} : G'_{a3} \sqsubseteq G_a$ and $c'_{b5} : G'_{b3} \sqsubseteq G_b$, we know that $(G_{a3}, c_{a3}, c_{a4}, c_{a5}) = \text{pullback}(G_a, (G_{a1}, c_{a1}), (G_{a2}, c_{a2}))$, $(G_{b3}, c_{b3}, c_{b4}, c_{b5}) = \text{pullback}(G_b, (G_{b1}, c_{b1}), (G_{b2}, c_{b2}))$ and there exists unique $c_a : G'_{a3} \sqsubseteq G_{a3}$ and $c_b : G'_{b3} \sqsubseteq G_{b3}$ such that

- $c_{a3} : G_{a3} \sqsubseteq G_{a1}$
- $c_{a4} : G_{a3} \sqsubseteq G_{a2}$
- $c_{a3};c_{a1} = c_{a5}$
- $c_{a4};c_{a2} = c_{a5}$
- $c_a;c_{a3} = c'_{a3}$
- $c_a;c_{a4} = c'_{a4}$
- $c_a;c_{a5} = c'_{a5}$
- $c_{b3} : G_{b3} \sqsubseteq G_{b1}$
- $c_{b4} : G_{b3} \sqsubseteq G_{b2}$
- $c_{b3};c_{b1} = c_{b5}$
- $c_{b4};c_{b2} = c_{b5}$
- $c_b;c_{b3} = c'_{b3}$
- $c_b;c_{b4} = c'_{b4}$
- $c_b;c_{b5} = c'_{b5}$

We know, by Definition of meet B.16, that

$$\begin{aligned} & \text{pullback}(G_a \longrightarrow G_b, (G_{a1} \longrightarrow G_{b1}, c_{a1} \longrightarrow c_{b1}), (G_{a2} \longrightarrow G_{b2}, c_{a2} \longrightarrow c_{b2})) = \\ & \quad (G_{c3} \longrightarrow G_{d3}, c_{c3} \longrightarrow c_{d3}, c_{c4} \longrightarrow c_{d4}, c_{c5} \longrightarrow c_{d5}) \mid \\ & (G_{c3}, c_{c3}, c_{c4}, c_{c5}) = \text{pullback}(G_a, (G_{a1}, c_{a1}), (G_{a2}, c_{a2})) \wedge (G_{d3}, c_{d3}, c_{d4}, c_{d5}) = \\ & \quad \text{pullback}(G_b, (G_{b1}, c_{b1}), (G_{b2}, c_{b2})) \end{aligned}$$

Therefore, by Definition of meet B.16,

$$\begin{aligned} & (G_{a3} \longrightarrow G_{b3}, c_{a3} \longrightarrow c_{b3}, c_{a4} \longrightarrow c_{b4}, c_{a5} \longrightarrow c_{b5}) \\ & = \text{pullback}(G_a \longrightarrow G_b, (G_{a1} \longrightarrow G_{b1}, c_{a1} \longrightarrow c_{b1}), (G_{a2} \longrightarrow G_{b2}, c_{a2} \longrightarrow c_{b2})) \end{aligned}$$

and there exists a unique $c_a \longrightarrow c_b : G'_{a3} \longrightarrow G'_{b3} \sqsubseteq G_{a3} \longrightarrow G_{b3}$ such that

- $c_{a3} \longrightarrow c_{b3} : G_{a3} \longrightarrow G_{b3} \sqsubseteq G_{a1} \longrightarrow G_{b1}$
- $c_{a4} \longrightarrow c_{b4} : G_{a3} \longrightarrow G_{b3} \sqsubseteq G_{a2} \longrightarrow G_{b2}$
- $c_{a3} \longrightarrow c_{b3};c_{a1} \longrightarrow c_{b1} = c_{a5} \longrightarrow c_{b5}$

- $\mathbf{c}_{a4} \longrightarrow \mathbf{c}_{b4}; \mathbf{c}_{a2} \longrightarrow \mathbf{c}_{b2} = \mathbf{c}_{a5} \longrightarrow \mathbf{c}_{b5}$
- $\mathbf{c}_a \longrightarrow \mathbf{c}_b; \mathbf{c}_{a3} \longrightarrow \mathbf{c}_{b3} = \mathbf{c}'_{a3} \longrightarrow \mathbf{c}'_{b3}$
- $\mathbf{c}_a \longrightarrow \mathbf{c}_b; \mathbf{c}_{a4} \longrightarrow \mathbf{c}_{b4} = \mathbf{c}'_{a4} \longrightarrow \mathbf{c}'_{b4}$
- $\mathbf{c}_a \longrightarrow \mathbf{c}_b; \mathbf{c}_{a5} \longrightarrow \mathbf{c}_{b5} = \mathbf{c}'_{a5} \longrightarrow \mathbf{c}'_{b5}$

Thus, the result holds.

5. $\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta} (\mathbf{c}'_5 : \mathbf{G}'_3 \sqsubseteq \mathbf{G})$. Then, $\mathbf{c}'_5 = \text{inj}_{\mathbf{B}}$, $\mathbf{G}'_3 = \mathbf{B}$ and $\mathbf{G} = ?_{\delta}$. Since $\mathbf{c}'_3; \mathbf{c}_1 = \mathbf{c}'_5$ and $\mathbf{c}'_4; \mathbf{c}_2 = \mathbf{c}'_5$, by Lemma B.12, we get the following cases.

- $\mathbf{c}'_3 = \mathbf{c}'_4 = \mathbf{B}$, $\mathbf{c}_1 = \mathbf{c}_2 = \text{inj}_{\mathbf{B}}$, $\mathbf{G}'_3 = \mathbf{G}_1 = \mathbf{G}_2 = \mathbf{B}$ and $\mathbf{G} = ?_{\delta}$. Therefore, we know that $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(?_{\delta}, (\mathbf{B}, \text{inj}_{\mathbf{B}}), (\mathbf{B}, \text{inj}_{\mathbf{B}})) = (\mathbf{B}, \mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{B}})$. Thus, we know that $(\mathbf{B}, \mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{B}}) = \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2))$ and there exists a unique $\mathbf{c} = \mathbf{B}$ such that $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$ ($\mathbf{c} : \mathbf{G}'_3 \sqsubseteq \mathbf{G}_3$) and
 - $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$ ($\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1$)
 - $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$ ($\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2$)
 - $\mathbf{B}; \text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$ ($\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5$)
 - $\mathbf{B}; \text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$ ($\mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}_5$)
 - $\mathbf{B}; \mathbf{B} = \mathbf{B}$ ($\mathbf{c}; \mathbf{c}_3 = \mathbf{c}'_3$)
 - $\mathbf{B}; \mathbf{B} = \mathbf{B}$ ($\mathbf{c}; \mathbf{c}_4 = \mathbf{c}'_4$)
 - $\mathbf{B}; \text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$ ($\mathbf{c}; \mathbf{c}_5 = \mathbf{c}'_5$)

Therefore, the result holds.

- $\mathbf{c}'_3 = \mathbf{B}$, $\mathbf{c}'_4 = \text{inj}_{\mathbf{B}}$, $\mathbf{c}_1 = \text{inj}_{\mathbf{B}}$, $\mathbf{c}_2 = \text{inj}_{?}$, $\mathbf{G}'_3 = \mathbf{B}$, $\mathbf{G}_1 = \mathbf{B}$, $\mathbf{G}_2 = ?_{\delta'}$ and $\mathbf{G} = ?_{\delta}$, where $\delta' \subseteq \delta$. Therefore, we know that $\text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = \text{pullback}(?_{\delta}, (?_{\delta'}, \text{inj}_{?}), (\mathbf{B}, \text{inj}_{\mathbf{B}})) = (\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B}, \text{inj}_{\mathbf{B}})$, and $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(?_{\delta}, (\mathbf{B}, \text{inj}_{\mathbf{B}}), (?_{\delta'}, \text{inj}_{?})) = (\mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{B}}, \text{inj}_{\mathbf{B}})$. Thus, we know that $(\mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{B}}, \text{inj}_{\mathbf{B}}) = \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2))$ and there exists a unique $\mathbf{c} = \mathbf{B}$ such that $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$ ($\mathbf{c} : \mathbf{G}'_3 \sqsubseteq \mathbf{G}_3$) and
 - $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$ ($\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1$)
 - $\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta'}$ ($\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2$)
 - $\mathbf{B}; \text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$ ($\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5$)
 - $\text{inj}_{\mathbf{B}}; \text{inj}_{?} = \text{inj}_{\mathbf{B}}$ ($\mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}_5$)
 - $\mathbf{B}; \mathbf{B} = \mathbf{B}$ ($\mathbf{c}; \mathbf{c}_3 = \mathbf{c}'_3$)
 - $\mathbf{B}; \text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$ ($\mathbf{c}; \mathbf{c}_4 = \mathbf{c}'_4$)
 - $\mathbf{B}; \text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$ ($\mathbf{c}; \mathbf{c}_5 = \mathbf{c}'_5$)

Therefore, the result holds.

- $\mathbf{c}'_3 = \text{inj}_{\mathbf{B}}$, $\mathbf{c}'_4 = \mathbf{B}$, $\mathbf{c}_1 = \text{inj}_{?}$, $\mathbf{c}_2 = \text{inj}_{\mathbf{B}}$, $\mathbf{G}'_3 = \mathbf{B}$, $\mathbf{G}_1 = ?_{\delta'}$, $\mathbf{G}_2 = \mathbf{B}$ and $\mathbf{G} = ?_{\delta}$, where $\delta' \in \delta$. Therefore, we know that $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(?_{\delta}, (?_{\delta'}, \text{inj}_{?}), (\mathbf{B}, \text{inj}_{\mathbf{B}})) = (\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B}, \text{inj}_{\mathbf{B}})$. Thus, we know that $(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B}, \text{inj}_{\mathbf{B}}) = \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2))$ and there exists a unique $\mathbf{c} = \mathbf{B}$ such that $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$ ($\mathbf{c} : \mathbf{G}'_3 \sqsubseteq \mathbf{G}_3$) and
 - $\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta'}$ ($\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1$)
 - $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$ ($\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2$)
 - $\text{inj}_{\mathbf{B}}; \text{inj}_{?} = \text{inj}_{\mathbf{B}}$ ($\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5$)
 - $\mathbf{B}; \text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$ ($\mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}_5$)

- $\mathbf{B};\text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}} (\mathbf{c};\mathbf{c}_3 = \mathbf{c}'_3)$
- $\mathbf{B};\mathbf{B} = \mathbf{B} (\mathbf{c};\mathbf{c}_4 = \mathbf{c}'_4)$
- $\mathbf{B};\text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}} (\mathbf{c};\mathbf{c}_5 = \mathbf{c}'_5)$

Therefore, the result holds.

- $\mathbf{c}'_3 = \mathbf{c}'_4 = \text{inj}_{\mathbf{B}}$, $\mathbf{c}_1 = \text{inj}_{\mathbf{?}}$, $\mathbf{c}_2 = \text{inj}_{\mathbf{?}}$, $\mathbf{G}'_3 = \mathbf{B}$, $\mathbf{G}_1 = \mathbf{?}_{\delta_1}$, $\mathbf{G}_2 = \mathbf{?}_{\delta_2}$ and $\mathbf{G} = \mathbf{?}_{\delta}$. Therefore, we know that $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\mathbf{?}_{\delta}, (\mathbf{?}_{\delta_1}, \text{inj}_{\mathbf{?}}), (\mathbf{?}_{\delta_2}, \text{inj}_{\mathbf{?}})) = (\mathbf{?}_{[\delta_1 \cap \delta_2]}, \text{inj}_{\mathbf{?}}, \text{inj}_{\mathbf{?}}, \text{inj}_{\mathbf{?}})$. Thus, we know that $(\mathbf{?}_{\delta_1 \cap \delta_2}, \text{inj}_{\mathbf{?}}, \text{inj}_{\mathbf{?}}, \text{inj}_{\mathbf{?}}) = \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2))$ and there exists a unique $\mathbf{c} = \text{inj}_{\mathbf{B}}$ such that $\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq \mathbf{?}_{\delta_1 \cap \delta_2} (\mathbf{c} : \mathbf{G}'_3 \sqsubseteq \mathbf{G}_3)$ and

- $\text{inj}_{\mathbf{?}} : \mathbf{?}_{\delta_1 \cap \delta_2} \sqsubseteq \mathbf{?}_{\delta_1} (\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1)$
- $\text{inj}_{\mathbf{?}} : \mathbf{?}_{\delta_1 \cap \delta_2} \sqsubseteq \mathbf{?}_{\delta_2} (\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2)$
- $\text{inj}_{\mathbf{?}}; \text{inj}_{\mathbf{?}} = \text{inj}_{\mathbf{?}} (\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5)$
- $\text{inj}_{\mathbf{?}}; \text{inj}_{\mathbf{?}} = \text{inj}_{\mathbf{?}} (\mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}_5)$
- $\text{inj}_{\mathbf{B}}; \text{inj}_{\mathbf{?}} = \text{inj}_{\mathbf{B}} (\mathbf{c}; \mathbf{c}_3 = \mathbf{c}'_3)$
- $\text{inj}_{\mathbf{B}}; \text{inj}_{\mathbf{?}} = \text{inj}_{\mathbf{B}} (\mathbf{c}; \mathbf{c}_4 = \mathbf{c}'_4)$
- $\text{inj}_{\mathbf{B}}; \text{inj}_{\mathbf{?}} = \text{inj}_{\mathbf{B}} (\mathbf{c}; \mathbf{c}_5 = \mathbf{c}'_5)$

Therefore, the result holds.

6. $\text{inj}_{\mathbf{X}} : \mathbf{F} \sqsubseteq \mathbf{?}_{\delta} (\mathbf{c}'_5 : \mathbf{G}'_3 \sqsubseteq \mathbf{G})$. Then, $\mathbf{c}'_5 = \text{inj}_{\mathbf{X}}$, $\mathbf{G}'_3 = \mathbf{F}$ and $\mathbf{G} = \mathbf{?}_{\delta}$, where $\mathbf{X} : \mathbf{F} \in \delta$. Since $\mathbf{c}'_3; \mathbf{c}_1 = \mathbf{c}'_5$ and $\mathbf{c}'_4; \mathbf{c}_2 = \mathbf{c}'_5$, by Lemma B.12, we get the following cases.

- $\mathbf{c}'_3 = \mathbf{c}'_4 = \mathbf{F}$, $\mathbf{c}_1 = \mathbf{c}_2 = \text{inj}_{\mathbf{X}}$, $\mathbf{G}'_3 = \mathbf{G}_1 = \mathbf{G}_2 = \mathbf{F}$ and $\mathbf{G} = \mathbf{?}_{\delta}$. Therefore, we know that $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\mathbf{?}_{\delta}, (\mathbf{F}, \text{inj}_{\mathbf{X}}), (\mathbf{F}, \text{inj}_{\mathbf{X}})) = (\mathbf{F}, \mathbf{F}, \mathbf{F}, \text{inj}_{\mathbf{X}})$ because $\mathbf{X} : \mathbf{F} \in \delta$. Thus, we know that $(\mathbf{F}, \mathbf{F}, \mathbf{F}, \text{inj}_{\mathbf{X}}) = \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2))$ and there exist a unique $\mathbf{c} = \mathbf{F}$ such that $\mathbf{F} : \mathbf{F} \sqsubseteq \mathbf{F} (\mathbf{c} : \mathbf{G}'_3 \sqsubseteq \mathbf{G}_3)$ and

- $\mathbf{F} : \mathbf{F} \sqsubseteq \mathbf{F} (\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1)$
- $\mathbf{F} : \mathbf{F} \sqsubseteq \mathbf{F} (\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2)$
- $\mathbf{F}; \text{inj}_{\mathbf{X}} = \text{inj}_{\mathbf{X}} (\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5)$
- $\mathbf{F}; \text{inj}_{\mathbf{X}} = \text{inj}_{\mathbf{X}} (\mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}_5)$
- $\mathbf{F}; \mathbf{F} = \mathbf{F} (\mathbf{c}; \mathbf{c}_3 = \mathbf{c}'_3)$
- $\mathbf{F}; \mathbf{F} = \mathbf{F} (\mathbf{c}; \mathbf{c}_4 = \mathbf{c}'_4)$
- $\mathbf{F}; \text{inj}_{\mathbf{X}} = \text{inj}_{\mathbf{X}} (\mathbf{c}; \mathbf{c}_5 = \mathbf{c}'_5)$

Therefore, the result holds.

- $\mathbf{c}'_3 = \mathbf{F}$, $\mathbf{c}'_4 = \text{inj}_{\mathbf{X}}$, $\mathbf{c}_1 = \text{inj}_{\mathbf{X}}$, $\mathbf{c}_2 = \text{inj}_{\mathbf{?}}$, $\mathbf{G}'_3 = \mathbf{F}$, $\mathbf{G}_1 = \mathbf{F}$, $\mathbf{G}_2 = \mathbf{?}_{\delta'}$ and $\mathbf{G} = \mathbf{?}_{\delta}$, where $\mathbf{X} : \mathbf{F} \in \delta' \subseteq \delta$. Therefore, we know that $\text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = \text{pullback}(\mathbf{?}_{\delta}, (\mathbf{?}_{\delta'}, \text{inj}_{\mathbf{?}}), (\mathbf{F}, \text{inj}_{\mathbf{X}})) = (\mathbf{F}, \text{inj}_{\mathbf{X}}, \mathbf{F}, \text{inj}_{\mathbf{X}})$ because $\mathbf{X} : \mathbf{F} \in \delta'$, and $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\mathbf{?}_{\delta}, (\mathbf{F}, \text{inj}_{\mathbf{X}}), (\mathbf{?}_{\delta'}, \text{inj}_{\mathbf{?}})) = (\mathbf{F}, \mathbf{F}, \text{inj}_{\mathbf{X}}, \text{inj}_{\mathbf{X}})$. Thus, we know that $(\mathbf{F}, \mathbf{F}, \text{inj}_{\mathbf{X}}, \text{inj}_{\mathbf{X}}) = \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2))$ and there exists a unique $\mathbf{c} = \mathbf{F}$ such that $\mathbf{F} : \mathbf{F} \sqsubseteq \mathbf{F} (\mathbf{c} : \mathbf{G}'_3 \sqsubseteq \mathbf{G}_3)$ and

- $\mathbf{F} : \mathbf{F} \sqsubseteq \mathbf{F} (\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1)$
- $\text{inj}_{\mathbf{X}} : \mathbf{F} \sqsubseteq \mathbf{?}_{\delta'} (\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2)$
- $\mathbf{F}; \text{inj}_{\mathbf{X}} = \text{inj}_{\mathbf{X}} (\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5)$
- $\text{inj}_{\mathbf{X}}; \text{inj}_{\mathbf{?}} = \text{inj}_{\mathbf{X}} (\mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}_5)$
- $\mathbf{F}; \mathbf{F} = \mathbf{F} (\mathbf{c}; \mathbf{c}_3 = \mathbf{c}'_3)$

- $\mathbf{F}; \text{inj}_{\mathbf{X}} = \text{inj}_{\mathbf{X}} (\mathbf{c}; \mathbf{c}_4 = \mathbf{c}'_4)$
- $\mathbf{F}; \text{inj}_{\mathbf{X}} = \text{inj}_{\mathbf{X}} (\mathbf{c}; \mathbf{c}_5 = \mathbf{c}'_5)$

Therefore, the result holds.

- $\mathbf{c}'_3 = \text{inj}_{\mathbf{X}}, \mathbf{c}'_4 = \mathbf{F}, \mathbf{c}_1 = \text{inj}_{\mathcal{?}}, \mathbf{c}_2 = \text{inj}_{\mathbf{X}}, \mathbf{G}'_3 = \mathbf{F}, \mathbf{G}_1 = \mathcal{?}_{\delta'}, \mathbf{G}_2 = \mathbf{F}$ and $\mathbf{G} = \mathcal{?}_{\delta}$, where $\mathbf{X} : \mathbf{F} \in \delta' \subseteq \delta$. Therefore, we know that $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\mathcal{?}_{\delta}, (\mathcal{?}_{\delta'}, \text{inj}_{\mathcal{?}}), (\mathbf{F}, \text{inj}_{\mathbf{X}})) = (\mathbf{F}, \text{inj}_{\mathbf{X}}, \mathbf{F}, \text{inj}_{\mathbf{X}})$ because of $\mathbf{X} : \mathbf{F} \in \delta'$. Thus, we know that $(\mathbf{F}, \text{inj}_{\mathbf{X}}, \mathbf{F}, \text{inj}_{\mathbf{X}}) = \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2))$ and there exists a unique $\mathbf{c} = \mathbf{F}$ such that $\mathbf{F} : \mathbf{F} \sqsubseteq \mathbf{F} (\mathbf{c} : \mathbf{G}'_3 \sqsubseteq \mathbf{G}_3)$ and

- $\text{inj}_{\mathbf{X}} : \mathbf{F} \sqsubseteq \mathcal{?}_{\delta'} (\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1)$
- $\mathbf{F} : \mathbf{F} \sqsubseteq \mathbf{F} (\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2)$
- $\text{inj}_{\mathbf{X}}; \text{inj}_{\mathcal{?}} = \text{inj}_{\mathbf{X}} (\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5)$
- $\mathbf{F}; \text{inj}_{\mathbf{X}} = \text{inj}_{\mathbf{X}} (\mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}_5)$
- $\mathbf{F}; \text{inj}_{\mathbf{X}} = \text{inj}_{\mathbf{X}} (\mathbf{c}; \mathbf{c}_3 = \mathbf{c}'_3)$
- $\mathbf{F}; \mathbf{F} = \mathbf{F} (\mathbf{c}; \mathbf{c}_4 = \mathbf{c}'_4)$
- $\mathbf{F}; \text{inj}_{\mathbf{X}} = \text{inj}_{\mathbf{X}} (\mathbf{c}; \mathbf{c}_5 = \mathbf{c}'_5)$

Therefore, the result holds.

- $\mathbf{c}'_3 = \mathbf{c}'_4 = \text{inj}_{\mathbf{X}}, \mathbf{c}_1 = \text{inj}_{\mathcal{?}}, \mathbf{c}_2 = \text{inj}_{\mathcal{?}}, \mathbf{G}'_3 = \mathbf{F}, \mathbf{G}_1 = \mathcal{?}_{\delta_1}, \mathbf{G}_2 = \mathcal{?}_{\delta_2}, \mathbf{X} : \mathbf{F} \in \delta_1, \mathbf{X} : \mathbf{F} \in \delta_2, \mathbf{X} : \mathbf{F} \in \delta_1 \cap \delta_2$, and $\mathbf{G} = \mathcal{?}_{\delta}$. Therefore, we know that $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\mathcal{?}_{\delta}, (\mathcal{?}_{\delta_1}, \text{inj}_{\mathcal{?}}), (\mathcal{?}_{\delta_2}, \text{inj}_{\mathcal{?}})) = (\mathcal{?}_{\delta_1 \cap \delta_2}, \text{inj}_{\mathcal{?}}, \text{inj}_{\mathcal{?}}, \text{inj}_{\mathcal{?}})$. Thus, we know that $(\mathcal{?}_{\delta_1 \cap \delta_2}, \text{inj}_{\mathcal{?}}, \text{inj}_{\mathcal{?}}, \text{inj}_{\mathcal{?}}) = \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2))$ and $\mathbf{c} = \text{inj}_{\mathbf{X}}$ such that $\text{inj}_{\mathbf{X}} : \mathbf{F} \sqsubseteq \mathcal{?}_{\delta_1 \cap \delta_2} (\mathbf{c} : \mathbf{G}'_3 \sqsubseteq \mathbf{G}_3)$, note that $\mathbf{X} : \mathbf{F} \in \delta_1 \cap \delta_2$ and

- $\text{inj}_{\mathcal{?}} : \mathcal{?}_{\delta_1 \cap \delta_2} \sqsubseteq \mathcal{?}_{\delta_1} (\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1)$
- $\text{inj}_{\mathcal{?}} : \mathcal{?}_{\delta_1 \cap \delta_2} \sqsubseteq \mathcal{?}_{\delta_2} (\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2)$
- $\text{inj}_{\mathcal{?}}; \text{inj}_{\mathcal{?}} = \text{inj}_{\mathcal{?}} (\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5)$
- $\text{inj}_{\mathcal{?}}; \text{inj}_{\mathcal{?}} = \text{inj}_{\mathcal{?}} (\mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}_5)$
- $\text{inj}_{\mathbf{X}}; \text{inj}_{\mathcal{?}} = \text{inj}_{\mathbf{X}} (\mathbf{c}; \mathbf{c}_3 = \mathbf{c}'_3)$
- $\text{inj}_{\mathbf{X}}; \text{inj}_{\mathcal{?}} = \text{inj}_{\mathbf{X}} (\mathbf{c}; \mathbf{c}_4 = \mathbf{c}'_4)$
- $\text{inj}_{\mathbf{X}}; \text{inj}_{\mathcal{?}} = \text{inj}_{\mathbf{X}} (\mathbf{c}; \mathbf{c}_5 = \mathbf{c}'_5)$

Therefore, the result holds.

7. $\text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}'_{a5}) : \forall \mathbf{X}. \mathbf{G}'_{a3} \sqsubseteq \mathcal{?}_{\delta} (\mathbf{c}'_5 : \mathbf{G}'_3 \sqsubseteq \mathbf{G})$. Then, $\mathbf{c}'_5 = \text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}'_{a5}), \mathbf{G}'_3 = \forall \mathbf{X}. \mathbf{G}'_{a3}$ and $\mathbf{G} = \mathcal{?}_{\delta}$. Since $\mathbf{c}'_3; \mathbf{c}_1 = \mathbf{c}'_5$ and $\mathbf{c}'_4; \mathbf{c}_2 = \mathbf{c}'_5$, by Lemma B.12, we get the following cases.

- $\mathbf{c}'_3 = \forall \mathbf{X}. \mathbf{c}'_{a3}, \mathbf{c}'_4 = \forall \mathbf{X}. \mathbf{c}'_{a4}, \mathbf{c}_1 = \text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}_{a1}), \mathbf{c}_2 = \text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}_{a2}), \mathbf{G}'_3 = \forall \mathbf{X}. \mathbf{G}'_{a3}, \mathbf{G}_1 = \forall \mathbf{X}. \mathbf{G}_{a1}, \mathbf{G}_2 = \forall \mathbf{X}. \mathbf{G}_{a2}$ and $\mathbf{G} = \mathcal{?}_{\delta}$. Therefore, we know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\mathcal{?}_{\delta}, (\forall \mathbf{X}. \mathbf{G}_{a1}, \text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}_{a1})), (\forall \mathbf{X}. \mathbf{G}_2, \text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}_{a2}))) \\ &= (\forall \mathbf{X}. \mathbf{G}_0, \forall \mathbf{X}. \mathbf{G}_0, \forall \mathbf{X}. \mathbf{G}_0, \text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}_0)) \end{aligned}$$

Thus, we know that $(\forall \mathbf{X}. \mathbf{G}_0, \forall \mathbf{X}. \mathbf{G}_0, \forall \mathbf{X}. \mathbf{G}_0, \text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}_0)) = \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2))$ and there exists a unique $\mathbf{c} = \forall \mathbf{X}. \mathbf{G}_0$ such that $\forall \mathbf{X}. \mathbf{G}_0 : \forall \mathbf{X}. \mathbf{G}_0 \sqsubseteq \forall \mathbf{X}. \mathbf{G}_0 (\mathbf{c} : \mathbf{G}'_3 \sqsubseteq \mathbf{G}_3)$ and

- $\forall \mathbf{X}. \mathbf{G}_0 : \forall \mathbf{X}. \mathbf{G}_0 \sqsubseteq \forall \mathbf{X}. \mathbf{G}_0 (\mathbf{c}_3 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_1)$
- $\forall \mathbf{X}. \mathbf{G}_0 : \forall \mathbf{X}. \mathbf{G}_0 \sqsubseteq \forall \mathbf{X}. \mathbf{G}_0 (\mathbf{c}_4 : \mathbf{G}_3 \sqsubseteq \mathbf{G}_2)$
- $\forall \mathbf{X}. \mathbf{G}_0; \text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}_0) = \text{inj}_{\forall}(\forall \mathbf{X}. \mathbf{c}_0) (\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}_5)$

- $\text{inj}_{\forall}(\forall X.c_0) : \forall X.G_0 \sqsubseteq ?_{\delta_2} (c_4 : G_3 \sqsubseteq G_2)$
- $\text{inj}_{\forall}(\forall X.c_0); \text{inj}_{?} = \text{inj}_{\forall}(\forall X.c_0) (c_3; c_1 = c_5)$
- $\text{inj}_{\forall}(\forall X.c_0); \text{inj}_{?} = \text{inj}_{\forall}(\forall X.c_0) (c_4; c_2 = c_5)$
- $\forall X.G_0; \text{inj}_{\forall}(\forall X.c_0) = \text{inj}_{\forall}(\forall X.c_0) (c; c_3 = c'_3)$
- $\forall X.G_0; \text{inj}_{\forall}(\forall X.c_0) = \text{inj}_{\forall}(\forall X.c_0) (c; c_4 = c'_4)$
- $\forall X.G_0; \text{inj}_{\forall}(\forall X.c_0) = \text{inj}_{\forall}(\forall X.c_0) (c; c_5 = c'_5)$

Therefore, the result holds.

8. $\text{inj}_{?} : ?_{\delta'} \sqsubseteq ?_{\delta} (c'_5 : G'_3 \sqsubseteq G)$. Then, $c'_5 = \text{inj}_{?}$, $G'_3 = ?_{\delta'}$ and $G = ?_{\delta}$. Since $c'_3; c_1 = c'_5$ and $c'_4; c_2 = c'_5$, by Lemma B.12, we know that $c'_3 = c_1 = c'_4 = c_2 = \text{inj}_{?}$, $G_1 = ?_{\delta_1}$ and $G_2 = ?_{\delta_2}$. Therefore, we know that $\text{pullback}(G, (G_1, c_1), (G_2, c_2)) = \text{pullback}(?_{\delta}, (?_{\delta_1}, \text{inj}_{?}), (?_{\delta_2}, \text{inj}_{?})) = (?_{\delta_1 \cap \delta_2}, \text{inj}_{?}, \text{inj}_{?}, \text{inj}_{?})$. Thus, we know that $\text{pullback}(G, (G_1, c_1), (G_2, c_2)) = (?_{\delta_1 \cap \delta_2}, \text{inj}_{?}, \text{inj}_{?}, \text{inj}_{?})$ and there exist a unique $c = \text{inj}_{?}$ such that $\text{inj}_{?} : ?_{\delta'} \sqsubseteq ?_{\delta} (c : G'_3 \sqsubseteq G_3)$ and

- $\text{inj}_{?} : ?_{\delta_1 \cap \delta_2} \sqsubseteq ?_{\delta_1} (c_3 : G_3 \sqsubseteq G_1)$
- $\text{inj}_{?} : ?_{\delta_1 \cap \delta_2} \sqsubseteq ?_{\delta_2} (c_4 : G_3 \sqsubseteq G_2)$
- $\text{inj}_{?}; \text{inj}_{?} = \text{inj}_{?} (c_3; c_1 = c_5)$
- $\text{inj}_{?}; \text{inj}_{?} = \text{inj}_{?} (c_4; c_2 = c_5)$
- $\text{inj}_{?}; \text{inj}_{?} = \text{inj}_{?} (c; c_3 = c'_3)$
- $\text{inj}_{?}; \text{inj}_{?} = \text{inj}_{?} (c; c_4 = c'_4)$
- $\text{inj}_{?}; \text{inj}_{?} = \text{inj}_{?} (c; c_5 = c'_5)$

Therefore, the result holds.

□

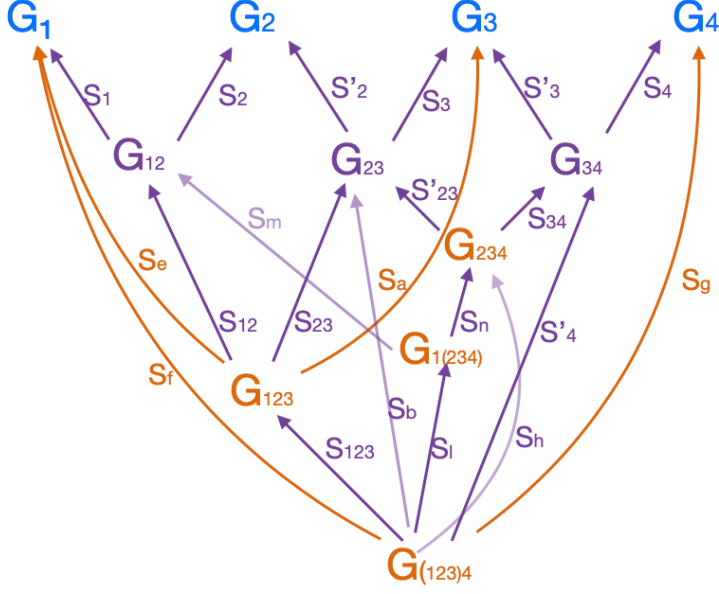
B.2.3 Associativity of consistent transitivity

Lemma B.18 (Associativity of Evidence composition) *Take*

- $\varepsilon_1 : G_1 \sim G_2$
- $\varepsilon_2 : G_2 \sim G_3$
- $\varepsilon_3 : G_3 \sim G_4$

Then

- $(\varepsilon_1 \circledast \varepsilon_2) \circledast \varepsilon_3 = \varepsilon_1 \circledast (\varepsilon_2 \circledast \varepsilon_3)$



PROOF. Let

- $(\mathbf{G}_{12}, \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon_1$
- $(\mathbf{G}_{23}, \mathbf{c}'_2, \mathbf{c}_3) \in \varepsilon_2$
- $(\mathbf{G}_{34}, \mathbf{c}'_3, \mathbf{c}_4) \in \varepsilon_3$

The evidence $(\varepsilon_1 \mathbin{\varepsilon} \varepsilon_2) \mathbin{\varepsilon} \varepsilon_3$ is then defined as illustrated in the following drawing:

We are required to prove that for all $(\mathbf{G}_{(123)4}, \mathbf{c}_f, \mathbf{c}_g)$, we get that $(\mathbf{G}_{(123)4}, \mathbf{c}_f, \mathbf{c}_g) \in ((\mathbf{G}_{12}, \mathbf{c}_1, \mathbf{c}_2) \mathbin{\varepsilon} (\mathbf{G}_{23}, \mathbf{c}'_2, \mathbf{c}_3)) \mathbin{\varepsilon} (\mathbf{G}_{34}, \mathbf{c}'_3, \mathbf{c}_4)$ if and only if $(\mathbf{G}_{(123)4}, \mathbf{c}_f, \mathbf{c}_g) \in (\mathbf{G}_{12}, \mathbf{c}_1, \mathbf{c}_2) \mathbin{\varepsilon} ((\mathbf{G}_{23}, \mathbf{c}'_2, \mathbf{c}_3) \mathbin{\varepsilon} (\mathbf{G}_{34}, \mathbf{c}'_3, \mathbf{c}_4))$. Let's suppose first that the evidence $(\mathbf{G}_{(123)4}, \mathbf{c}_f, \mathbf{c}_g) \in ((\mathbf{G}_{12}, \mathbf{c}_1, \mathbf{c}_2) \mathbin{\varepsilon} (\mathbf{G}_{23}, \mathbf{c}'_2, \mathbf{c}_3)) \mathbin{\varepsilon} (\mathbf{G}_{34}, \mathbf{c}'_3, \mathbf{c}_4)$. Then, we are required to prove that $(\mathbf{G}_{(123)4}, \mathbf{c}_f, \mathbf{c}_g) \in (\mathbf{G}_{12}, \mathbf{c}_1, \mathbf{c}_2) \mathbin{\varepsilon} ((\mathbf{G}_{23}, \mathbf{c}'_2, \mathbf{c}_3) \mathbin{\varepsilon} (\mathbf{G}_{34}, \mathbf{c}'_3, \mathbf{c}_4))$ (the backwards direction is similar).

A Therefore, we know that $(\mathbf{G}_{12}, \mathbf{c}_1, \mathbf{c}_2) \mathbin{\varepsilon} (\mathbf{G}_{23}, \mathbf{c}'_2, \mathbf{c}_3)$ is defined. Thus, we know that $\text{pullback}(\mathbf{G}_2, (\mathbf{G}_{12}, \mathbf{c}_2), (\mathbf{G}_{23}, \mathbf{c}'_2))$ is defined. Therefore, $\text{pullback}(\mathbf{G}_2, (\mathbf{G}_{12}, \mathbf{c}_2), (\mathbf{G}_{23}, \mathbf{c}'_2)) = (\mathbf{G}_{123}, \mathbf{c}_{12}, \mathbf{c}_{23}, \mathbf{c}_0)$, and we know by definition of meet (Definition B.16) that

- A.1) $\mathbf{c}_{12}; \mathbf{c}_2 = \mathbf{c}_0$
- A.2) $\mathbf{c}_{23}; \mathbf{c}'_2 = \mathbf{c}_0$
- A.3) $\mathbf{c}_{12} : \mathbf{G}_{123} \sqsubseteq \mathbf{G}_{12}$
- A.4) $\mathbf{c}_{23} : \mathbf{G}_{123} \sqsubseteq \mathbf{G}_{23}$
- A.5) $\mathbf{c}_0 : \mathbf{G}_{123} \sqsubseteq \mathbf{G}_2$
- A.6) $\mathbf{c}_2 : \mathbf{G}_{12} \sqsubseteq \mathbf{G}_2$
- A.7) $\mathbf{c}'_2 : \mathbf{G}_{23} \sqsubseteq \mathbf{G}_2$

B Then, we know that there exists $(\mathbf{G}_{123}, \mathbf{c}_e, \mathbf{c}_a) \in (\mathbf{G}_{12}, \mathbf{c}_1, \mathbf{c}_2) \mathbin{\varepsilon} (\mathbf{G}_{23}, \mathbf{c}'_2, \mathbf{c}_3)$ such that (by definition of consistent transitivity)

- B.1) $\mathbf{c}_{23};\mathbf{c}_3 = \mathbf{c}_a$
- B.2) $\mathbf{c}_{12};\mathbf{c}_1 = \mathbf{c}_e$
- B.3) $\mathbf{c}_a : \mathbf{G}_{123} \sqsubseteq \mathbf{G}_3$
- B.4) $\mathbf{c}_e : \mathbf{G}_{123} \sqsubseteq \mathbf{G}_1$
- B.5) $\mathbf{c}_1 : \mathbf{G}_{12} \sqsubseteq \mathbf{G}_1$
- B.6) $\mathbf{c}_3 : \mathbf{G}_{23} \sqsubseteq \mathbf{G}_3$

C Also, we know that $\text{pullback}(\mathbf{G}_3, (\mathbf{G}_{123}, \mathbf{c}_a), (\mathbf{G}_{34}, \mathbf{c}'_3))$ is defined. Therefore, $\text{pullback}(\mathbf{G}_3, (\mathbf{G}_{123}, \mathbf{c}_a), (\mathbf{G}_{34}, \mathbf{c}'_3)) = (\mathbf{G}_{(123)4}, \mathbf{c}_{123}, \mathbf{c}'_4, \mathbf{c}_c)$, and we know by definition of meet (Definition B.16) that

- C.1) $\mathbf{c}_{123};\mathbf{c}_a = \mathbf{c}_c$
- C.2) $\mathbf{c}'_4;\mathbf{c}'_3 = \mathbf{c}_c$
- C.3) $\mathbf{c}_{123} : \mathbf{G}_{(123)4} \sqsubseteq \mathbf{G}_{123}$
- C.4) $\mathbf{c}'_4 : \mathbf{G}_{(123)4} \sqsubseteq \mathbf{G}_{34}$
- C.5) $\mathbf{c}_c : \mathbf{G}_{(123)4} \sqsubseteq \mathbf{G}_3$
- C.6) $\mathbf{c}'_3 : \mathbf{G}_{34} \sqsubseteq \mathbf{G}_3$

D Thus, $(\mathbf{G}_{(123)4}, \mathbf{c}_f, \mathbf{c}_g) \in (\mathbf{G}_{123}, \mathbf{c}_e, \mathbf{c}_a) \circ (\mathbf{G}_{34}, \mathbf{c}'_3, \mathbf{c}_4)$ and (by definition of consistent transitivity)

- D.1) $\mathbf{c}_{123};\mathbf{c}_e = \mathbf{c}_f$
- D.2) $\mathbf{c}_4;\mathbf{c}'_4 = \mathbf{c}_g$
- D.3) $\mathbf{c}_f : \mathbf{G}_{(123)4} \sqsubseteq \mathbf{G}_1$
- D.4) $\mathbf{c}_g : \mathbf{G}_{(123)4} \sqsubseteq \mathbf{G}_4$
- D.5) $\mathbf{c}_4 : \mathbf{G}_{34} \sqsubseteq \mathbf{G}_4$

E By (B.1) and (C.1), we get that $\mathbf{c}_{123};(\mathbf{c}_{23};\mathbf{c}_3) = \mathbf{c}_c$, and by Lemma B.13, we get that $(\mathbf{c}_{123};\mathbf{c}_{23});\mathbf{c}_3 = \mathbf{c}_c$. By Definition B.21, we get that there exists \mathbf{c}_b such that

- E.1) $\mathbf{c}_{123};\mathbf{c}_{23} = \mathbf{c}_b$
- E.2) $\mathbf{c}_b;\mathbf{c}_3 = \mathbf{c}_c$
- E.3) $\mathbf{c}_b : \mathbf{G}_{(123)4} \sqsubseteq \mathbf{G}_{23}$

F By (B.2) and (D.1), we get that $\mathbf{c}_{123};(\mathbf{c}_{12};\mathbf{c}_1) = \mathbf{c}_f$, and by Lemma B.13, we get that $(\mathbf{c}_{123};\mathbf{c}_{12});\mathbf{c}_1 = \mathbf{c}_f$. By Definition B.21, we get that there exists \mathbf{c}_p such that

- F.1) $\mathbf{c}_{123};\mathbf{c}_{12} = \mathbf{c}_p$
- F.2) $\mathbf{c}_p;\mathbf{c}_1 = \mathbf{c}_f$
- F.3) $\mathbf{c}_p : \mathbf{G}_{(123)4} \sqsubseteq \mathbf{G}_{12}$

G By (B.6), (C.6), (E.3), (C.4), (E.2) and (C.2), we make use of the universal property (Lemma 6.3), this means that we must have $\text{pullback}(\mathbf{G}_3, (\mathbf{G}_{23}, \mathbf{c}_3), (\mathbf{G}_{34}, \mathbf{c}'_3)) = (\mathbf{G}_{234}, \mathbf{c}'_{23}, \mathbf{c}_{34}, \mathbf{c}'_o)$ (i.e. this meet must be defined) and there exists a unique \mathbf{c}_h such that

- G.1) $\mathbf{c}_h;\mathbf{c}_{34} = \mathbf{c}'_4$

$$\mathbf{G.2)} \quad \mathbf{c}_h; \mathbf{c}'_{23} = \mathbf{c}_b$$

$$\mathbf{G.3)} \quad \mathbf{c}_h; \mathbf{c}'_o = \mathbf{c}_c$$

$$\mathbf{G.4)} \quad \mathbf{c}'_{23}; \mathbf{c}_3 = \mathbf{c}'_o$$

$$\mathbf{G.5)} \quad \mathbf{c}_{34}; \mathbf{c}'_3 = \mathbf{c}'_o$$

$$\mathbf{G.6)} \quad \mathbf{c}_h : \mathbf{G}_{(123)4} \sqsubseteq \mathbf{G}_{234}$$

H By (C.3), (A.5) and Lemma B.20, we get that there exists \mathbf{c}_k such that $\mathbf{c}_k : \mathbf{G}_{(123)4} \sqsubseteq \mathbf{G}_2$ and $\mathbf{c}_{123}; \mathbf{c}_o = \mathbf{c}_k$. By $\mathbf{c}_{123}; \mathbf{c}_o = \mathbf{c}_k$, (A.1) and (F.1), and, (A.2) and (E.1), and Lemma B.13, we get that

$$\mathbf{H.1)} \quad \mathbf{c}_p; \mathbf{c}_2 = \mathbf{c}_k$$

$$\mathbf{H.2)} \quad \mathbf{c}_b; \mathbf{c}'_2 = \mathbf{c}_k$$

$$\mathbf{H.3)} \quad \mathbf{c}_{123}; \mathbf{c}_o = \mathbf{c}_k$$

$$\mathbf{H.4)} \quad \mathbf{c}_k : \mathbf{G}_{(123)4} \sqsubseteq \mathbf{G}_2$$

I By (G.2) and (H.2), we get that $(\mathbf{c}_h; \mathbf{c}'_{23}); \mathbf{c}'_2 = \mathbf{c}_k$, and by Lemma B.13, we get that $\mathbf{c}_h; (\mathbf{c}'_{23}; \mathbf{c}'_2) = \mathbf{c}_k$. By Definition B.21, we get that there exists \mathbf{c}_i such that

$$\mathbf{I.1)} \quad \mathbf{c}'_{23}; \mathbf{c}'_2 = \mathbf{c}_i$$

$$\mathbf{I.2)} \quad \mathbf{c}_h; \mathbf{c}_i = \mathbf{c}_k$$

$$\mathbf{I.3)} \quad \mathbf{c}_i : \mathbf{G}_{234} \sqsubseteq \mathbf{G}_2$$

J By (G.1) and (D.2), we get that $(\mathbf{c}_h; \mathbf{c}_{34}); \mathbf{c}_4 = \mathbf{c}_g$, and by Lemma B.13, we get that $\mathbf{c}_h; (\mathbf{c}_{34}; \mathbf{c}_4) = \mathbf{c}_g$. By Definition B.21, we get that there exists \mathbf{c}'_i such that

$$\mathbf{J.1)} \quad \mathbf{c}_{34}; \mathbf{c}_4 = \mathbf{c}'_i$$

$$\mathbf{J.2)} \quad \mathbf{c}_h; \mathbf{c}'_i = \mathbf{c}_g$$

$$\mathbf{J.3)} \quad \mathbf{c}'_i : \mathbf{G}_{234} \sqsubseteq \mathbf{G}_4$$

Note that $(\mathbf{G}_{234}, \mathbf{c}_i, \mathbf{c}'_i) \in (\mathbf{G}_{23}, \mathbf{c}'_2, \mathbf{c}_3) \circ (\mathbf{G}_{34}, \mathbf{c}'_3, \mathbf{c}_4)$, by the definition of consistent transitivity.

K By (A.6), (I.3), (F.3), (G.6), (H.1) and (I.2), we make use of the universal property (Lemma 6.3), this means that we must have $\text{pullback}(\mathbf{G}_2, (\mathbf{G}_{12}, \mathbf{c}_2), (\mathbf{G}_{234}, \mathbf{c}_i)) = (\mathbf{G}_{1(234)}, \mathbf{c}_m, \mathbf{c}_n, \mathbf{c}''_o)$ (i.e. this meet must be defined) and there exists a unique \mathbf{c}_1 such that

$$\mathbf{K.1)} \quad \mathbf{c}_1; \mathbf{c}_m = \mathbf{c}_p$$

$$\mathbf{K.2)} \quad \mathbf{c}_1; \mathbf{c}_n = \mathbf{c}_h$$

$$\mathbf{K.3)} \quad \mathbf{c}_1; \mathbf{c}''_o = \mathbf{c}_k$$

$$\mathbf{K.4)} \quad \mathbf{c}_m; \mathbf{c}_2 = \mathbf{c}''_o$$

$$\mathbf{K.5)} \quad \mathbf{c}_n; \mathbf{c}_i = \mathbf{c}''_o$$

$$\mathbf{K.6)} \quad \mathbf{c}_m : \mathbf{G}_{1(234)} \sqsubseteq \mathbf{G}_{12}$$

$$\mathbf{K.7)} \quad \mathbf{c}_n : \mathbf{G}_{1(234)} \sqsubseteq \mathbf{G}_{234}$$

$$\mathbf{K.8)} \quad \mathbf{c}''_o : \mathbf{G}_{1(234)} \sqsubseteq \mathbf{G}_2$$

$$\mathbf{K.9)} \quad \mathbf{c}_1 : \mathbf{G}_{(123)4} \sqsubseteq \mathbf{G}_{1(234)}$$

L By (K.2) and (G.2), we get that $(c_1; c_n); c'_{23} = c_b$, and by Lemma B.13, we get that $c_1; (c_n; c'_{23}) = c_b$. By Definition B.21, we get that there exists c'_b such that

$$\text{L.1) } c_n; c'_{23} = c'_b$$

$$\text{L.2) } c_1; c'_b = c_b$$

$$\text{L.3) } c'_b : G_{1(234)} \sqsubseteq G_{23}$$

M By (I.1) and (K.5), we know that $c_n; (c_{23}; c'_2) = c''_o$, and by Lemma B.13, we get that $(c_n; c_{23}); c'_2 = c''_o$. Then, by (L.1), we get that

$$\text{M.1) } c'_b; c'_2 = c''_o$$

N By (A.6), (A.7), (K.6), (L.3), (K.4) and (M.1), we make use of the universal property (Lemma 6.3), this means that there exists a unique c'_h such that (we already know that $\text{pullback}(G_2, (G_{12}, c_2), (G_{23}, c'_2)) = (G_{123}, c_{12}, c_{23}, c_o)$)

$$\text{N.1) } c'_h; c_{12} = c_m$$

$$\text{N.2) } c'_h; c_{23} = c'_b$$

$$\text{N.3) } c'_h; c_o = c''_o$$

$$\text{N.4) } c_{12}; c_2 = c_o$$

$$\text{N.5) } c_{23}; c'_2 = c_o$$

$$\text{N.6) } c'_h : G_{1(234)} \sqsubseteq G_{123}$$

O By (N.1) and (K.1), we know that $c_1; (c'_h; c_{12}) = c_p$, and by Lemma B.13, we get that $(c_1; c'_h); c_{12} = c_p$. Since $c_{123}; c_{12} = c_p$, we get that

$$\text{O.1) } c_1; c'_h = c_{123}$$

P By (K.2) and (G.3), we get that $(c_1; c_n); c'_o = c_c$, and by Lemma B.13, we get that $c_1; (c_n; c'_o) = c_c$. By Definition B.21, we get that there exists c'_c such that

$$\text{P.1) } c_n; c'_o = c'_c$$

$$\text{P.2) } c_1; c'_c = c_c$$

$$\text{P.3) } c'_c : G_{1(234)} \sqsubseteq G_3$$

Q By (K.2) and (G.1), we get that $(c_1; c_n); c_{34} = c'_4$, and by Lemma B.13, we get that $c_1; (c_n; c_{34}) = c'_4$. By Definition B.21, we get that there exists c''_4 such that

$$\text{Q.1) } c_n; c_{34} = c''_4$$

$$\text{Q.2) } c_1; c''_4 = c'_4$$

$$\text{Q.3) } c''_4 : G_{1(234)} \sqsubseteq G_{34}$$

R By (C.2) and (Q.2), we know that $(c_1; c''_4); c'_3 = c_c$, and by Lemma B.13, we get that $c_1; (c''_4; c'_3) = c_c$. Since (P.2), we get that

$$\text{R.1) } c''_4; c'_3 = c'_c$$

S By (C.1) and (O.1), we know that $(c_1; c'_h); c_a = c_c$, and by Lemma B.13, we get that $c_1; (c'_h; c_a) = c_c$. Since (P.2), we get that

$$\text{S.1) } c'_h; c_a = c'_c$$

T By (B.6), (C.6), (N.6), (Q.3), (S.1) and (R.1), we make use of the universal property (Lemma 6.3), this means that there exists \mathbf{c}'_1 such that (we already know that $\text{pullback}(\mathbf{G}_3, (\mathbf{G}_{123}, \mathbf{c}_a), (\mathbf{G}_{34}, \mathbf{c}'_3)) = (\mathbf{G}_{(123)4}, \mathbf{c}_{123}, \mathbf{c}'_4, \mathbf{c}_c)$)

T.1) $\mathbf{c}'_1; \mathbf{c}_{123} = \mathbf{c}'_h$

T.2) $\mathbf{c}'_1; \mathbf{c}'_4 = \mathbf{c}''_4$

T.3) $\mathbf{c}'_1; \mathbf{c}_c = \mathbf{c}'_c$

T.4) $\mathbf{c}_{123}; \mathbf{c}_3 = \mathbf{c}_c$

T.5) $\mathbf{c}'_4; \mathbf{c}'_3 = \mathbf{c}_c$

T.6) $\mathbf{c}'_1 : \mathbf{G}_{1(234)} \sqsubseteq \mathbf{G}_{(123)4}$

U Since (T.6) and (K.9), we can conclude from Lemma B.22 that $\mathbf{c}'_1 = \mathbf{c}_1 = \text{refl}_{\mathbf{G}_{(123)4}}^{\sqsubseteq}$ and $\mathbf{G}_{1(234)} = \mathbf{G}_{(123)4}$.

V By (K.1) and (F.2), we get that $(\mathbf{c}_1; \mathbf{c}_m); \mathbf{c}_1 = \mathbf{c}_f$. By Lemma B.13, we know that $\mathbf{c}_1; (\mathbf{c}_m; \mathbf{c}_1) = \mathbf{c}_f$. By Definition B.21, there exists \mathbf{c}_f^* such that $\mathbf{c}_m; \mathbf{c}_1 = \mathbf{c}_f^*$ and $\mathbf{c}_1; \mathbf{c}_f^* = \mathbf{c}_f$. By Lemma B.14, $\mathbf{c}_1 = \text{refl}_{\mathbf{G}_{(123)4}}^{\sqsubseteq}$ and $\mathbf{c}_1; \mathbf{c}_f^* = \mathbf{c}_f$, we know that $\mathbf{c}_f^* = \mathbf{c}_f$ and $\mathbf{c}_m; \mathbf{c}_1 = \mathbf{c}_f$.

W By (J.2) and (K.2), we get that $(\mathbf{c}_1; \mathbf{c}_n); \mathbf{c}'_i = \mathbf{c}_g$. By Lemma B.13, we know that $\mathbf{c}_1; (\mathbf{c}_n; \mathbf{c}'_i) = \mathbf{c}_g$. By Definition B.21, there exists \mathbf{c}_g^* such that $\mathbf{c}_n; \mathbf{c}'_i = \mathbf{c}_g^*$ and $\mathbf{c}_1; \mathbf{c}_g^* = \mathbf{c}_g$. By Lemma B.14, $\mathbf{c}_1 = \text{refl}_{\mathbf{G}_{(123)4}}^{\sqsubseteq}$ and $\mathbf{c}_1; \mathbf{c}_g^* = \mathbf{c}_g$, we know that $\mathbf{c}_g^* = \mathbf{c}_g$ and $\mathbf{c}_n; \mathbf{c}'_i = \mathbf{c}_g$.

X Finally, the result holds, since $(\mathbf{G}_{1(234)}, \mathbf{c}_f, \mathbf{c}_g) \in (\mathbf{G}_{12}, \mathbf{c}_1, \mathbf{c}_2) \ddagger (\mathbf{G}_{234}, \mathbf{c}_i, \mathbf{c}'_i)$, by definition of consistent transitivity.

□

Lemma B.19 *If*

- $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$
- $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}_3$
- $\mathbf{c}_1; \mathbf{c}_2 = \mathbf{c}_3$

then

- $\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$

Note that this lemma doesn't follow from Lemma B.20, because this lemma says that $\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$ must hold for all \mathbf{c}_3 , not just the one that is shown to exist in that other lemma.

PROOF. We proceed by induction on $\mathbf{c}_1; \mathbf{c}_2 = \mathbf{c}_3$.

- $\mathbf{B}; \mathbf{B} = \mathbf{B}$. Then, we know that $\mathbf{c}_1 = \mathbf{c}_2 = \mathbf{c}_3 = \mathbf{B}$ and $\mathbf{G}_1 = \mathbf{G}_2 = \mathbf{G}_3 = \mathbf{B}$. Therefore, $\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$ because $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$, and the result holds.

- $\mathbf{X};\mathbf{X} = \mathbf{X}$. Then, we know that $\mathbf{c}_1 = \mathbf{c}_2 = \mathbf{c}_3 = \mathbf{X}$ and $\mathbf{G}_1 = \mathbf{G}_2 = \mathbf{G}_3 = \mathbf{X}$. Therefore, $\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$ because $\mathbf{X} : \mathbf{X} \sqsubseteq \mathbf{X}$, and the result holds.
- $\forall \mathbf{X}.\mathbf{c}_1; \forall \mathbf{X}.\mathbf{c}'_2 = \forall \mathbf{X}.\mathbf{c}'_3$, where $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'_3$. Then, we know that $\mathbf{c}_1 = \forall \mathbf{X}.\mathbf{c}_1$, $\mathbf{c}_2 = \forall \mathbf{X}.\mathbf{c}'_2$, $\mathbf{c}_3 = \forall \mathbf{X}.\mathbf{c}'_3$, $\mathbf{G}_1 = \forall \mathbf{X}.\mathbf{G}'_1$, $\mathbf{G}_2 = \forall \mathbf{X}.\mathbf{G}'_2$ and $\mathbf{G}_3 = \forall \mathbf{X}.\mathbf{G}'_3$, where $\mathbf{c}_1 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$, $\mathbf{c}'_2 : \mathbf{G}'_2 \sqsubseteq \mathbf{G}'_3$ and $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'_3$. Then, by the induction hypothesis on $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'_3$, we know that $\mathbf{c}'_2 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_3$. Therefore, $\forall \mathbf{X}.\mathbf{c}'_3 : \forall \mathbf{X}.\mathbf{G}'_1 \sqsubseteq \forall \mathbf{X}.\mathbf{G}'_3$ ($\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$), and the result holds.
- $\mathbf{c}_1 \longrightarrow \mathbf{c}''_1; \mathbf{c}'_2 \longrightarrow \mathbf{c}''_2 = \mathbf{c}'_3 \longrightarrow \mathbf{c}''_3$, where $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'_3$ and $\mathbf{c}''_1; \mathbf{c}''_2 = \mathbf{c}''_3$. Then, we know that $\mathbf{c}_1 = \mathbf{c}_1 \longrightarrow \mathbf{c}''_1$, $\mathbf{c}_2 = \mathbf{c}'_2 \longrightarrow \mathbf{c}''_2$, $\mathbf{c}_3 = \mathbf{c}'_3 \longrightarrow \mathbf{c}''_3$, $\mathbf{G}_1 = \mathbf{G}'_1 \longrightarrow \mathbf{G}''_1$, $\mathbf{G}_2 = \mathbf{G}'_2 \longrightarrow \mathbf{G}''_2$ and $\mathbf{G}_3 = \mathbf{G}'_3 \longrightarrow \mathbf{G}''_3$, where $\mathbf{c}_1 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$, $\mathbf{c}''_1 : \mathbf{G}''_1 \sqsubseteq \mathbf{G}''_2$, $\mathbf{c}'_2 : \mathbf{G}'_2 \sqsubseteq \mathbf{G}'_3$, $\mathbf{c}''_2 : \mathbf{G}''_2 \sqsubseteq \mathbf{G}''_3$, $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'_3$ and $\mathbf{c}''_1; \mathbf{c}''_2 = \mathbf{c}''_3$. Then, by the induction hypothesis on $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'_3$ and $\mathbf{c}''_1; \mathbf{c}''_2 = \mathbf{c}''_3$, we know that $\mathbf{c}'_2 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_3$ and $\mathbf{c}''_2 : \mathbf{G}''_1 \sqsubseteq \mathbf{G}''_3$. Therefore, $\mathbf{c}'_2 \longrightarrow \mathbf{c}''_2 : \mathbf{G}'_1 \longrightarrow \mathbf{G}''_1 \sqsubseteq \mathbf{G}'_3 \longrightarrow \mathbf{G}''_3$ ($\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$), and the result holds.
- $\mathbf{B}; \text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$. Then, we know that $\mathbf{c}_1 = \mathbf{B}$, $\mathbf{c}_2 = \mathbf{c}_3 = \text{inj}_{\mathbf{B}}$, $\mathbf{G}_1 = \mathbf{G}_2 = \mathbf{B}$ and $\mathbf{G}_3 = ?_{\delta}$. Therefore, $\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$ because $\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta}$, and the result holds.
- $\mathbf{F}; \text{inj}_{\mathbf{X}} = \text{inj}_{\mathbf{X}}$. Then, we know that $\mathbf{c}_1 = \mathbf{F}$, $\mathbf{c}_2 = \mathbf{c}_3 = \text{inj}_{\mathbf{X}}$, $\mathbf{G}_1 = \mathbf{G}_2 = \mathbf{F}$ and $\mathbf{G}_3 = ?_{\delta}$, where $\mathbf{X} : \mathbf{F} \in \delta$. Therefore, $\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$ because $\text{inj}_{\mathbf{X}} : \mathbf{F} \sqsubseteq ?_{\delta}$, and the result holds.
- $\mathbf{c}_1; \text{inj}_{\rightarrow}(\mathbf{c}'_2) = \text{inj}_{\rightarrow}(\mathbf{c}'_3)$, where $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'_3$. Inspecting the type precision rules, we have the following cases. Then, we know that $\mathbf{c}_2 = \text{inj}_{\rightarrow}(\mathbf{c}'_2)$, $\mathbf{c}_3 = \text{inj}_{\rightarrow}(\mathbf{c}'_3)$ and $\mathbf{G}_3 = ?_{\delta}$. Inspecting the type precision rules, we have the following cases.
 - $\mathbf{c}_2 = \text{inj}_{\forall}(\mathbf{c}'_2)$ and $\mathbf{c}_3 = \text{inj}_{\forall}(\mathbf{c}'_3)$. Therefore, $\mathbf{c}'_2 : \mathbf{G}_2 \sqsubseteq \forall \mathbf{X}?._{\delta, \mathbf{X}:\mathbf{X}}$. Thus, by the induction hypothesis on $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'_3$, we get that $\mathbf{c}'_3 : \mathbf{G}_1 \sqsubseteq \forall \mathbf{X}?._{\delta, \mathbf{X}:\mathbf{X}}$. Thus, $\text{inj}_{\forall}(\mathbf{c}'_3) : \mathbf{G}_1 \sqsubseteq ?_{\delta}$. Therefore, $\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$, and the result holds.
 - $\mathbf{c}_2 = \text{inj}_{\rightarrow}(\mathbf{c}'_2)$ and $\mathbf{c}_3 = \text{inj}_{\rightarrow}(\mathbf{c}'_3)$. Therefore, $\mathbf{c}'_2 : \mathbf{G}_2 \sqsubseteq ?_{\delta} \longrightarrow ?_{\delta}$. Thus, by the induction hypothesis on $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'_3$, we get that $\mathbf{c}'_3 : \mathbf{G}_1 \sqsubseteq ?_{\delta} \longrightarrow ?_{\delta}$. Thus, $\text{inj}_{\rightarrow}(\mathbf{c}'_3) : \mathbf{G}_1 \sqsubseteq ?_{\delta}$. Therefore, $\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$, and the result holds.
- $\mathbf{c}; \text{inj}_{?} = \mathbf{c}$. Then, we know that $\mathbf{c}_1 = \mathbf{c}_3 = \mathbf{c}$, $\mathbf{c}_2 = \text{inj}_{?}$, $\mathbf{G}_2 = ?_{\delta}$ and $\mathbf{G}_3 = ?_{\delta', \overline{\mathbf{X}:\mathbf{F}}}$, where $\delta \subseteq \delta'$. Since $\mathbf{c} : \mathbf{G}_1 \sqsubseteq ?_{\delta}$, by Lemma B.15, we know that $\mathbf{c} : \mathbf{G}_1 \sqsubseteq ?_{\delta', \overline{\mathbf{X}:\mathbf{F}}}$. Therefore, $\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$ because $\mathbf{c} : \mathbf{G}_1 \sqsubseteq ?_{\delta', \overline{\mathbf{X}:\mathbf{F}}}$, and the result holds.

□

Lemma B.20 *If*

- $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$
- $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}_3$

then there exists a \mathbf{c}_3 such that

- $\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$
- $\mathbf{c}_1; \mathbf{c}_2 = \mathbf{c}_3$

PROOF. We proceed by induction on $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}_3$.

Case ($\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$). Since $\mathbf{G}_2 = \mathbf{B}$, by Lemma B.11, we know that $\mathbf{c}_1 = \mathbf{B}$ and $\mathbf{G}_1 = \mathbf{B}$. Since, $\mathbf{B};\mathbf{B} = \mathbf{B}$, the result holds.

Case ($\mathbf{X} : \mathbf{X} \sqsubseteq \mathbf{X}$). Since $\mathbf{G}_2 = \mathbf{X}$, by Lemma B.11, we know that $\mathbf{c}_1 = \mathbf{X}$ and $\mathbf{G}_1 = \mathbf{X}$. Since, $\mathbf{X};\mathbf{X} = \mathbf{X}$, the result holds.

Case ($\mathbf{c}_{21} \rightarrow \mathbf{c}_{22} : \mathbf{G}_{21} \rightarrow \mathbf{G}_{22} \sqsubseteq \mathbf{G}_{31} \rightarrow \mathbf{G}_{32}$). Since $\mathbf{G}_2 = \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}$, by Lemma B.11, we know that $\mathbf{c}_1 = \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}$ and $\mathbf{G}_1 = \mathbf{G}_{11} \rightarrow \mathbf{G}_{12}$. We know that $\mathbf{c}_{21} : \mathbf{G}_{21} \sqsubseteq \mathbf{G}_{31}$, $\mathbf{c}_{22} : \mathbf{G}_{22} \sqsubseteq \mathbf{G}_{32}$, $\mathbf{c}_{11} : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{c}_{12} : \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$. By the inductive hypothesis, we get that $\mathbf{c}_{11};\mathbf{c}_{21} = \mathbf{c}_{31}$ and $\mathbf{c}_{12};\mathbf{c}_{22} = \mathbf{c}_{32}$. Therefore, we get that $\mathbf{c}_{11} \rightarrow \mathbf{c}_{12}; \mathbf{c}_{21} \rightarrow \mathbf{c}_{22} = \mathbf{c}_{31} \rightarrow \mathbf{c}_{32}$, and the result holds.

Case ($\mathbf{c}_{21} \rightarrow \mathbf{c}_{22} : \mathbf{G}_{21} \rightarrow \mathbf{G}_{22} \sqsubseteq \mathbf{G}_{31} \rightarrow \mathbf{G}_{32}$). Since $\mathbf{G}_2 = \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}$, by Lemma B.11, we know that $\mathbf{c}_1 = \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}$ and $\mathbf{G}_1 = \mathbf{G}_{11} \rightarrow \mathbf{G}_{12}$. We know that $\mathbf{c}_{21} : \mathbf{G}_{21} \sqsubseteq \mathbf{G}_{31}$, $\mathbf{c}_{22} : \mathbf{G}_{22} \sqsubseteq \mathbf{G}_{32}$, $\mathbf{c}_{11} : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{c}_{12} : \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$. By the inductive hypothesis, we get that $\mathbf{c}_{11};\mathbf{c}_{21} = \mathbf{c}_{31}$ and $\mathbf{c}_{12};\mathbf{c}_{22} = \mathbf{c}_{32}$. Therefore, we get that $\mathbf{c}_{11} \rightarrow \mathbf{c}_{12}; \mathbf{c}_{21} \rightarrow \mathbf{c}_{22} = \mathbf{c}_{31} \rightarrow \mathbf{c}_{32}$, and the result holds.

Case ($\mathbf{c}_{21} \times \mathbf{c}_{22} : \mathbf{G}_{21} \times \mathbf{G}_{22} \sqsubseteq \mathbf{G}_{31} \times \mathbf{G}_{32}$). Since $\mathbf{G}_2 = \mathbf{G}_{21} \times \mathbf{G}_{22}$, by Lemma B.11, we know that $\mathbf{c}_1 = \mathbf{c}_{11} \times \mathbf{c}_{12}$ and $\mathbf{G}_1 = \mathbf{G}_{11} \times \mathbf{G}_{12}$. We know that $\mathbf{c}_{21} : \mathbf{G}_{21} \sqsubseteq \mathbf{G}_{31}$, $\mathbf{c}_{22} : \mathbf{G}_{22} \sqsubseteq \mathbf{G}_{32}$, $\mathbf{c}_{11} : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{c}_{12} : \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$. By the inductive hypothesis, we get that $\mathbf{c}_{11};\mathbf{c}_{21} = \mathbf{c}_{31}$ and $\mathbf{c}_{12};\mathbf{c}_{22} = \mathbf{c}_{32}$. Therefore, we get that $\mathbf{c}_{11} \times \mathbf{c}_{12}; \mathbf{c}_{21} \times \mathbf{c}_{22} = \mathbf{c}_{31} \times \mathbf{c}_{32}$, and the result holds.

Case ($\forall \mathbf{X}. \mathbf{c}_{21} : \forall \mathbf{X}. \mathbf{G}_{21} \sqsubseteq \forall \mathbf{X}. \mathbf{G}_{31}$). Since $\mathbf{G}_2 = \forall \mathbf{X}. \mathbf{G}_{21}$, by Lemma B.11, we know that $\mathbf{c}_1 = \forall \mathbf{X}. \mathbf{c}_{11}$ and $\mathbf{G}_1 = \forall \mathbf{X}. \mathbf{G}_{11}$. We know that $\mathbf{c}_{21} : \mathbf{G}_{21} \sqsubseteq \mathbf{G}_{31}$ and $\mathbf{c}_{11} : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$. By the inductive hypothesis, we get that $\mathbf{c}_{11};\mathbf{c}_{21} = \mathbf{c}_{31}$. Therefore, we get that $\forall \mathbf{X}. \mathbf{c}_{11}; \forall \mathbf{X}. \mathbf{c}_{21} = \forall \mathbf{X}. \mathbf{c}_{31}$, and the result holds.

Case ($\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta}$). Since $\mathbf{G}_2 = \mathbf{B}$, by Lemma B.11, we know that $\mathbf{c}_1 = \mathbf{B}$ and $\mathbf{G}_1 = \mathbf{B}$. Since, $\mathbf{B};\text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$, the result holds.

Case ($\text{inj}_{\mathbf{X}} : \mathbf{F} \sqsubseteq ?_{\delta} \wedge \mathbf{X} : \mathbf{F} \in \delta$). Since $\mathbf{G}_2 = \mathbf{F}$, by Lemma B.11, we know that $\mathbf{c}_1 = \mathbf{F}$ and $\mathbf{G}_1 = \mathbf{F}$. Since, $\mathbf{F};\text{inj}_{\mathbf{X}} = \text{inj}_{\mathbf{X}}$, the result holds.

Case ($\text{inj}_{\rightarrow}(\mathbf{c}'_2) : \mathbf{G}_2 \sqsubseteq ?_{\delta}$). We know that $\mathbf{c}'_2 : \mathbf{G}_2 \sqsubseteq ?_{\delta} \rightarrow ?_{\delta}$ and $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$. Thus, by the inductive hypothesis, we get that $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'_3$. Therefore, $\mathbf{c}_1; \text{inj}_{\rightarrow}(\mathbf{c}'_2) = \text{inj}_{\rightarrow}(\mathbf{c}'_3)$, and the result holds immediately.

Case ($\text{inj}_{\times}(\mathbf{c}'_2) : \mathbf{G}_2 \sqsubseteq ?_{\delta}$). We know that $\mathbf{c}'_2 : \mathbf{G}_2 \sqsubseteq ?_{\delta} \times ?_{\delta}$ and $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$. Thus, by the inductive hypothesis, we get that $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'_3$. Therefore, $\mathbf{c}_1; \text{inj}_{\times}(\mathbf{c}'_2) = \text{inj}_{\times}(\mathbf{c}'_3)$, and the result holds immediately.

Case ($\text{inj}_{\vee}(\mathbf{c}'_2) : \mathbf{G}_2 \sqsubseteq ?_{\delta}$). We know that $\mathbf{c}'_2 : \mathbf{G}_2 \sqsubseteq \forall \mathbf{X}. ?_{\delta}, \mathbf{X} : \mathbf{X}$ and $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$. Thus, by the inductive hypothesis, we get that $\mathbf{c}_1; \mathbf{c}'_2 = \mathbf{c}'_3$. Therefore, $\mathbf{c}_1; \text{inj}_{\vee}(\mathbf{c}'_2) = \text{inj}_{\vee}(\mathbf{c}'_3)$, and the result holds immediately.

Case ($\text{inj}_{?} : ?_{\delta} \sqsubseteq ?_{\delta'}$). We know that $\mathbf{c}_1; \text{inj}_{?} = \mathbf{c}_1$. Since $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq ?_{\delta}$, by Lemma B.15, we get that $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq ?_{\delta'}$, and the result holds.

□

Definition B.21 We write $(\mathbf{c}_1; \mathbf{c}_2); \mathbf{c}_3 = \mathbf{c}_4$ to mean that there exists \mathbf{c}' such that $\mathbf{c}_1; \mathbf{c}_2 = \mathbf{c}'$ and $\mathbf{c}'; \mathbf{c}_3 = \mathbf{c}_4$. In the same way, we write $\mathbf{c}_1; (\mathbf{c}_2; \mathbf{c}_3) = \mathbf{c}_4$ to mean that there exists \mathbf{c}' such that $\mathbf{c}_2; \mathbf{c}_3 = \mathbf{c}'$ and $\mathbf{c}_1; \mathbf{c}' = \mathbf{c}_4$.

Lemma B.22 (Antisymmetry) *If*

- $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$

- $\mathbf{c}_2 : \mathbf{G}_2 \sqsubseteq \mathbf{G}_1$

Then $\mathbf{G}_1 = \mathbf{G}_2$ and $\mathbf{c}_1 = \text{refl}_{\mathbf{G}_1}^{\sqsubseteq} = \mathbf{c}_2$.

PROOF. By induction on the derivation of $\mathbf{c}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$. Most of the cases follow easily by inversion and induction.

Case $(\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B})$. By Lemma B.11, we know that $\mathbf{c}_2 = \mathbf{B} = \text{refl}_{\mathbf{B}}^{\sqsubseteq} = \mathbf{c}_1$ and $\mathbf{G}_1 = \mathbf{G}_2$. Thus, the result holds.

Case $(\mathbf{X} : \mathbf{X} \sqsubseteq \mathbf{X})$. By Lemma B.11, we know that $\mathbf{c}_2 = \mathbf{X} = \text{refl}_{\mathbf{X}}^{\sqsubseteq} = \mathbf{c}_1$ and $\mathbf{G}_1 = \mathbf{G}_2$. Thus, the result holds.

Case $(\mathbf{c}_{11} \rightarrow \mathbf{c}'_{11} : \mathbf{G}_{11} \rightarrow \mathbf{G}'_{11} \sqsubseteq \mathbf{G}_{21} \rightarrow \mathbf{G}'_{21})$. We know that $\mathbf{c}_{11} : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{c}'_{11} : \mathbf{G}'_{11} \sqsubseteq \mathbf{G}'_{21}$. By Lemma B.11, we know that $\mathbf{c}_2 = \mathbf{c}_{22} \rightarrow \mathbf{c}'_{22}$, where $\mathbf{c}_{22} : \mathbf{G}_{21} \sqsubseteq \mathbf{G}_{11}$ and $\mathbf{c}'_{22} : \mathbf{G}'_{21} \sqsubseteq \mathbf{G}'_{11}$. By the inductive hypothesis, we get that $\mathbf{c}_{11} = \mathbf{c}_{22} = \text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq}$, $\mathbf{G}_{11} = \mathbf{G}_{21}$, $\mathbf{c}'_{11} = \mathbf{c}'_{22} = \text{refl}_{\mathbf{G}'_{11}}^{\sqsubseteq}$ and $\mathbf{G}'_{11} = \mathbf{G}'_{21}$, as we are required to show.

Case $(\mathbf{c}_{11} \times \mathbf{c}'_{11} : \mathbf{G}_{11} \times \mathbf{G}'_{11} \sqsubseteq \mathbf{G}_{21} \times \mathbf{G}'_{21})$. We know that $\mathbf{c}_{11} : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{c}'_{11} : \mathbf{G}'_{11} \sqsubseteq \mathbf{G}'_{21}$. By Lemma B.11, we know that $\mathbf{c}_2 = \mathbf{c}_{22} \times \mathbf{c}'_{22}$, where $\mathbf{c}_{22} : \mathbf{G}_{21} \sqsubseteq \mathbf{G}_{11}$ and $\mathbf{c}'_{22} : \mathbf{G}'_{21} \sqsubseteq \mathbf{G}'_{11}$. By the inductive hypothesis, we get that $\mathbf{c}_{11} = \mathbf{c}_{22} = \text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq}$, $\mathbf{G}_{11} = \mathbf{G}_{21}$, $\mathbf{c}'_{11} = \mathbf{c}'_{22} = \text{refl}_{\mathbf{G}'_{11}}^{\sqsubseteq}$ and $\mathbf{G}'_{11} = \mathbf{G}'_{21}$, as we are required to show.

Case $(\forall \mathbf{X}. \mathbf{c}_{11} : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21})$. We know that $\mathbf{c}_{11} : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$. By Lemma B.11, we know that $\mathbf{c}_2 = \forall \mathbf{X}. \mathbf{c}_{22}$, where $\mathbf{c}_{22} : \mathbf{G}_{21} \sqsubseteq \mathbf{G}_{11}$. By the inductive hypothesis, we get that $\mathbf{c}_{11} = \mathbf{c}_{22} = \text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq}$ and $\mathbf{G}_{11} = \mathbf{G}_{21}$, as we are required to show.

Case $(\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta})$. This case is not possible because is not the case that $?_{\delta} \sqsubseteq \mathbf{B}$.

Case $(\text{inj}_{\mathbf{X}} : \mathbf{F} \sqsubseteq ?_{\delta} \wedge \mathbf{X} : \mathbf{F} \in \delta)$. This case is not possible because is not the case that $?_{\delta} \sqsubseteq \mathbf{F}$.

Case $(\text{inj}_{\rightarrow}(\mathbf{c}) : \mathbf{G} \sqsubseteq ?_{\delta})$. This case is not possible because is not the case that $?_{\delta} \sqsubseteq \mathbf{G}$ (note that $\mathbf{c} : \mathbf{G} \sqsubseteq ?_{\delta} \rightarrow ?_{\delta}$).

Case $(\text{inj}_{\times}(\mathbf{c}) : \mathbf{G} \sqsubseteq ?_{\delta})$. This case is not possible because is not the case that $?_{\delta} \sqsubseteq \mathbf{G}$ (note that $\mathbf{c} : \mathbf{G} \sqsubseteq ?_{\delta} \times ?_{\delta}$).

Case $(\text{inj}_{\vee}(\mathbf{c}) : \mathbf{G} \sqsubseteq ?_{\delta})$. This case is not possible because is not the case that $?_{\delta} \sqsubseteq \mathbf{G}$ (note that $\mathbf{c} : \mathbf{G} \sqsubseteq \forall \mathbf{X}. ?_{\delta, \mathbf{X} : \mathbf{X}}$).

Case $(\text{inj}_{?} : ?_{\delta} \sqsubseteq ?_{\delta'} \wedge \delta \subseteq \delta')$. By Lemma B.11, we know that $\mathbf{c}_2 = \text{inj}_{?}$ and $\delta' \subseteq \delta$. Since $\delta' \subseteq \delta$ and $\delta \subseteq \delta'$, we know that $\delta = \delta'$, and therefore $\mathbf{G}_1 = \mathbf{G}_2 = ?_{\delta} = ?_{\delta'}$ and $\mathbf{c}_1 = \mathbf{c}_2 = \text{inj}_{?} = \text{refl}_{?_{\delta}}^{\sqsubseteq}$, as we are required to prove.

□

B.2.4 Evidence operations

Definition B.23 *If*

- $\varepsilon_1 = \{(\mathbf{G}_1, \mathbf{c}_{11}, \mathbf{c}_{12})\}$
- $\varepsilon_1 : \mathbf{G}_{11} \sim \mathbf{G}_{12}$
- $\varepsilon_2 = \{(\mathbf{G}_2, \mathbf{c}_{21}, \mathbf{c}_{22})\}$

- $\varepsilon_2 : \mathbf{G}_{21} \sim \mathbf{G}_{22}$

Then we define

- $\varepsilon_1 \longrightarrow \varepsilon_2 = \{ (\mathbf{G}_1 \rightarrow \mathbf{G}_2, \mathbf{c}_{11} \longrightarrow \mathbf{c}_{21}, \mathbf{c}_{12} \longrightarrow \mathbf{c}_{22}) \}$
- $\varepsilon_1 \times \varepsilon_2 = \{ (\mathbf{G}_1 \times \mathbf{G}_2, \mathbf{c}_{11} \times \mathbf{c}_{21}, \mathbf{c}_{12} \times \mathbf{c}_{22}) \}$
- $\forall \mathbf{X}.\varepsilon_1 = \{ (\forall \mathbf{X}.\mathbf{G}_1, \forall \mathbf{X}.\mathbf{c}_{11}, \forall \mathbf{X}.\mathbf{c}_{12}) \}$
- $\text{reflEv}(\mathbf{B}) = \{ (\mathbf{B}, \mathbf{B}, \mathbf{B}) \}$
- $\text{reflEv}(\mathbf{X}) = \{ (\mathbf{X}, \mathbf{X}, \mathbf{X}) \}$
- $\text{reflEv}(\delta) = \{ (\delta, \text{inj}_?, \text{inj}_?) \}$

and we have:

- $(\varepsilon_1 \longrightarrow \varepsilon_2) : (\mathbf{G}_{11} \longrightarrow \mathbf{G}_{21}) \sim (\mathbf{G}_{12} \longrightarrow \mathbf{G}_{22})$
- $(\varepsilon_1 \times \varepsilon_2) : (\mathbf{G}_{11} \times \mathbf{G}_{21}) \sim (\mathbf{G}_{12} \times \mathbf{G}_{22})$
- $(\forall \mathbf{X}.\varepsilon_1) : (\forall \mathbf{X}.\mathbf{G}_{11}) \sim (\forall \mathbf{X}.\mathbf{G}_{12})$
- $\text{reflEv}(\mathbf{B}) : \mathbf{B} \sim \mathbf{B}$
- $\text{reflEv}(\delta) : \delta \sim \delta$
- $\text{reflEv}(\mathbf{X}) : \mathbf{X} \sim \mathbf{X}$

Definition B.24 For any gradual type \mathbf{G} , we define $\text{refl}_\mathbf{G}^\sqsubseteq : \mathbf{G} \sqsubseteq \mathbf{G}$:

$$\begin{aligned} \text{refl}_\mathbf{B}^\sqsubseteq &= \mathbf{B} & \text{refl}_\mathbf{X}^\sqsubseteq &= \mathbf{X} & \text{refl}_{\mathbf{G}_1 \rightarrow \mathbf{G}_2}^\sqsubseteq &= \text{refl}_{\mathbf{G}_1}^\sqsubseteq \longrightarrow \text{refl}_{\mathbf{G}_2}^\sqsubseteq \\ \text{refl}_{\mathbf{G}_1 \times \mathbf{G}_2}^\sqsubseteq &= \text{refl}_{\mathbf{G}_1}^\sqsubseteq \times \text{refl}_{\mathbf{G}_2}^\sqsubseteq & \text{refl}_{\forall \mathbf{X}.\mathbf{G}}^\sqsubseteq &= \forall \mathbf{X}.\text{refl}_\mathbf{G}^\sqsubseteq & \text{refl}_{\delta}^\sqsubseteq &= \text{inj}_? \end{aligned}$$

Definition B.25 (Reflexive Evidence) For any gradual type \mathbf{G} , we extend the definition of $\text{reflEv}(\mathbf{B})$, $\text{reflEv}(\mathbf{X})$ and $\text{reflEv}(\delta)$ above with the following definitions:

$$\begin{aligned} \text{reflEv}(\mathbf{G}_1 \rightarrow \mathbf{G}_2) &= \text{reflEv}(\mathbf{G}_1) \longrightarrow \text{reflEv}(\mathbf{G}_2) \\ \text{reflEv}(\mathbf{G}_1 \times \mathbf{G}_2) &= \text{reflEv}(\mathbf{G}_1) \times \text{reflEv}(\mathbf{G}_2) \\ \text{reflEv}(\forall \mathbf{X}.\mathbf{G}) &= \forall \mathbf{X}.\text{reflEv}(\mathbf{G}) \end{aligned}$$

Then we have that $\text{reflEv}(\mathbf{G}) : \mathbf{G} \sqsubseteq \mathbf{G}$.

Definition B.26 (Inject) If $\text{ftv}(\mathbf{G}) \subseteq \text{dom}(\delta)$, then we define $\text{inject}_\mathbf{G}^\sqsubseteq : \mathbf{G} \sqsubseteq \delta$:

$$\begin{aligned} \text{inject}_\mathbf{B}^\sqsubseteq &= \text{inj}_\mathbf{B} \\ \text{inject}_\mathbf{X}^\sqsubseteq &= \text{inj}_\mathbf{X} \\ \text{inject}_{\delta}^\sqsubseteq &= \text{inj}_? \\ \text{inject}_{\mathbf{G}_1 \rightarrow \mathbf{G}_2}^\sqsubseteq &= \text{inj}_\rightarrow(\text{inject}_{\mathbf{G}_1}^\sqsubseteq \rightarrow \text{inject}_{\mathbf{G}_2}^\sqsubseteq) \\ \text{inject}_{\mathbf{G}_1 \times \mathbf{G}_2}^\sqsubseteq &= \text{inj}_\times(\text{inject}_{\mathbf{G}_1}^\sqsubseteq \times \text{inject}_{\mathbf{G}_2}^\sqsubseteq) \\ \text{inject}_{\forall \mathbf{X}.\mathbf{G}}^\sqsubseteq &= \text{inj}_\forall(\forall \mathbf{X}.\text{inject}_\mathbf{G}^\sqsubseteq) \end{aligned}$$

B.3 $F_{\varepsilon}^?$: Gradual Parametricity

Example. We illustrate the interpretation of the unknown type by considering some examples. Let $\varepsilon_{\text{Int}^?} = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\text{Int}})\}$, $\varepsilon_{\text{Int}} = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{refl}_{\text{Int}})\}$, $\varepsilon_{\text{X}^?} = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\text{X}})\}$, $\delta = \text{X} : \text{X}$, and $\delta' = \text{X} : \text{Int}$.

- $(n, \varepsilon_{\text{Int}^?}42 :: ?_{\delta'}, \varepsilon_{\text{Int}^?}42 :: ?_{\delta'}) \in \mathcal{V}_{\rho}[[?_{\delta}]]$ because for $\varepsilon = \{(\text{Int}, \text{inj}_{\text{Int}}, \text{refl}_{\text{Int}})\} \vdash \varepsilon : \delta \rightarrow \text{Int}$, as $\varepsilon_{\text{Int}^?} \circ \rho.i(\varepsilon) = \varepsilon_{\text{Int}}$, then $\triangleright (n, \varepsilon_{\text{Int}}42 :: \text{Int}, \varepsilon_{\text{Int}}42 :: \text{Int}) \in \mathcal{V}_{\rho}[[\text{Int}]]$ (and for every other evidence consistent transitivity is not defined).
- $(n, \varepsilon_{\text{Int}^?}42 :: ?_{\delta'}, \varepsilon_{\text{Int}^?}43 :: ?_{\delta'}) \notin \mathcal{V}_{\rho}[[?_{\delta}]]$ because for $\vdash \{(\text{Int}, \text{inj}_{\text{Int}}, \text{refl}_{\text{Int}})\} : \delta \rightarrow \text{Int}$, as $\triangleright (n, \varepsilon_{\text{Int}}42 :: \text{Int}, \varepsilon_{\text{Int}}43 :: \text{Int}) \notin \mathcal{V}_{\rho}[[\text{Int}]]$.
- Suppose $\triangleright (n, \varepsilon_{\text{Int}}42 :: \text{Int}, \varepsilon_{\text{Int}}43 :: \text{Int}) \in \rho.R(X)$. Then $(n, \varepsilon_{\text{X}^?}42 :: ?_{\delta'}, \varepsilon_{\text{X}^?}43 :: ?_{\delta'}) \in \mathcal{V}_{\rho}[[?_{\delta}]]$ because for $\varepsilon = \{(\text{X}, \text{inj}_{\text{X}}, \text{refl}_{\text{X}})\}$, $\vdash \varepsilon : \delta \rightarrow \text{X}$, as $\varepsilon_{\text{X}^?} \circ \rho.i(\varepsilon) = \varepsilon_{\text{Int}}$, then $\triangleright (n, \varepsilon_{\text{Int}}42 :: \text{Int}, \varepsilon_{\text{Int}}43 :: \text{Int}) \in \mathcal{V}_{\rho}[[\text{X}]] = \rho.R(X)$.
- But $(n, \varepsilon_{\text{X}^?}42 :: ?_{\delta'}, \varepsilon_{\text{X}^?}43 :: ?_{\delta'}) \notin \mathcal{V}_{\rho}[[?_{\delta'}]]$ because for $\varepsilon = \{(\text{Int}, \text{inj}_{\text{X}}, \text{refl}_{\text{Int}})\} \vdash \varepsilon : \delta' \rightarrow \text{Int}$, as $\varepsilon_{\text{X}^?} \circ \rho.i(\varepsilon) = \varepsilon_{\text{Int}}$, but $\triangleright (n, \varepsilon_{\text{Int}}42 :: \text{Int}, \varepsilon_{\text{Int}}43 :: \text{Int}) \notin \mathcal{V}_{\rho}[[\text{Int}]]$.
- Suppose $\rho.R(X) = \mathcal{V}_{\rho}[[\text{Int}]]$, and $\varepsilon_{\text{m}} = \varepsilon_{\text{Int}^?} \cup \varepsilon_{\text{X}^?}$. Then $(n, \varepsilon_{\text{m}}42 :: ?_{\delta}, \varepsilon_{\text{m}}42 :: ?_{\delta}) \in \mathcal{V}_{\rho}[[?_{\delta}]]$ because (1) for $\varepsilon = \{(\text{X}, \text{inj}_{\text{X}}, \text{refl}_{\text{X}})\}$, $\vdash \varepsilon : \delta \rightarrow \text{X}$, as $\varepsilon_{\text{m}} \circ \rho.i(\varepsilon) = \varepsilon_{\text{X}^?} \circ \rho.i(\varepsilon) = \varepsilon_{\text{Int}}$, then $\triangleright (n, \varepsilon_{\text{Int}}42 :: \text{Int}, \varepsilon_{\text{Int}}42 :: \text{Int}) \in \mathcal{V}_{\rho}[[\text{X}]] = \rho.R(X)$; and (2) for $\varepsilon = \{(\text{Int}, \text{inj}_{\text{Int}}, \text{refl}_{\text{Int}})\}$, $\vdash \varepsilon : \delta \rightarrow \text{Int}$, as $\varepsilon_{\text{m}} \circ \rho.i(\varepsilon) = \varepsilon_{\text{Int}^?} \circ \rho.i(\varepsilon) = \varepsilon_{\text{Int}}$, then $\triangleright (n, \varepsilon_{\text{Int}}42 :: \text{Int}, \varepsilon_{\text{Int}}42 :: \text{Int}) \in \mathcal{V}_{\rho}[[\text{Int}]]$.

Theorem B.27 (Fundamental Property) *If $\Xi; \Delta; \Gamma \vdash t : G$ then $\Xi; \Delta; \Gamma \vdash t \preceq t : G$.*

PROOF. By induction on the type derivation of \mathbf{t} .

Case (**Gasc**). Then $\mathbf{t} = \varepsilon \mathbf{s} :: \mathbf{G}$, and therefore:

$$(\text{Gasc}) \frac{\Delta; \Gamma \vdash \mathbf{s} : \mathbf{G}' \quad \varepsilon : \mathbf{G}' \sim \mathbf{G}}{\Delta; \Gamma \vdash \varepsilon \mathbf{s} :: \mathbf{G} : \mathbf{G}}$$

We follow by induction on the structure of \mathbf{s} .

- If $\mathbf{s} = \mathbf{b}$ then:

$$(\text{Gb}) \frac{\theta(\mathbf{b}) = \mathbf{B} \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \mathbf{b} : \mathbf{B}}$$

Then we have to prove that $\Delta; \Gamma \vdash \varepsilon \mathbf{b} :: \mathbf{G} \preceq \varepsilon \mathbf{b} :: \mathbf{G} : \mathbf{G}$, but the result follows directly by Prop B.28 (Compatibility of Constant).

- If $\mathbf{s} = \lambda \mathbf{x} : \mathbf{G}_1. \mathbf{t}'$ then:

$$(\text{G}\lambda) \frac{\Delta; \Gamma, \mathbf{x} : \mathbf{G}_1 \vdash \mathbf{t}' : \mathbf{G}_2}{\Delta; \Gamma \vdash \lambda \mathbf{x} : \mathbf{G}_1. \mathbf{t}' : \mathbf{G}_1 \rightarrow \mathbf{G}_2}$$

Then we have to prove that

$$\Delta; \Gamma \vdash \varepsilon (\lambda \mathbf{x} : \mathbf{G}_1. \mathbf{t}') :: \mathbf{G} \preceq \varepsilon (\lambda \mathbf{x} : \mathbf{G}_1. \mathbf{t}') :: \mathbf{G} : \mathbf{G}$$

Since $\Delta; \Gamma, \mathbf{x} : \mathbf{G}_1 \vdash \mathbf{t}' : \mathbf{G}_2$, by the induction hypotheses we know that $\Delta; \Gamma, \mathbf{x} : \mathbf{G}_1 \vdash \mathbf{t}' \preceq \mathbf{t}' : \mathbf{G}_2$. Then, the result follows directly by Prop B.29 (Compatibility of term abstraction).

- If $\mathbf{s} = \Lambda X. \mathbf{t}'$ then:

$$(\mathbf{G}\Lambda) \frac{\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}' : \mathbf{G}'' \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \Lambda X. \mathbf{t}' : \forall X. \mathbf{G}''}$$

Then we have to prove that

$$\Delta; \Gamma \vdash \varepsilon(\Lambda X. \mathbf{t}') :: \mathbf{G} \preceq \varepsilon(\Lambda X. \mathbf{t}') :: \mathbf{G} : \mathbf{G}$$

Since $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}' : \mathbf{G}''$, by the induction hypotheses we know that $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}' \preceq \mathbf{t}' : \mathbf{G}''$. Then, the result follows directly by Prop B.30 (Compatibility of type abstraction).

- If $\mathbf{s} = \langle \mathbf{u}_1, \mathbf{u}_2 \rangle$ then:

$$(\mathbf{G}\text{pair}) \frac{\Delta; \Gamma \vdash \mathbf{u}_1 : \mathbf{G}_1 \quad \Delta; \Gamma \vdash \mathbf{u}_2 : \mathbf{G}_2}{\Delta; \Gamma \vdash \langle \mathbf{u}_1, \mathbf{u}_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2}$$

Then we have to prove that

$$\Delta; \Gamma \vdash \varepsilon \langle \mathbf{u}_1, \mathbf{u}_2 \rangle :: \mathbf{G} \preceq \varepsilon \langle \mathbf{u}_1, \mathbf{u}_2 \rangle :: \mathbf{G} : \mathbf{G}$$

Since $\varepsilon : \mathbf{G}_1 \times \mathbf{G}_2 \sim \mathbf{G}$, we know that \mathbf{G} can be either $\mathbf{G}'_1 \times \mathbf{G}'_2$ or $?_\delta$.

- $\mathbf{G} = \mathbf{G}'_1 \times \mathbf{G}'_2$. We know by premise that $\Delta; \Gamma \vdash \pi_1(\varepsilon) \mathbf{u}_1 :: \mathbf{G}'_1 : \mathbf{G}'_1$ and $\Delta; \Gamma \vdash \pi_2(\varepsilon) \mathbf{u}_2 :: \mathbf{G}'_2 : \mathbf{G}'_2$. Then by the induction hypotheses we know that $\Delta; \Gamma \vdash \pi_1(\varepsilon) \mathbf{u}_1 :: \mathbf{G}'_1 \preceq \pi_1(\varepsilon) \mathbf{u}_1 :: \mathbf{G}'_1 : \mathbf{G}'_1$ and $\Delta; \Gamma \vdash \pi_2(\varepsilon) \mathbf{u}_2 :: \mathbf{G}'_2 \preceq \pi_2(\varepsilon) \mathbf{u}_2 :: \mathbf{G}'_2 : \mathbf{G}'_2$. Then, the result follows directly by Prop B.31 (Compatibility of pairs).
- $\mathbf{G} = ?_\delta$. By Lemma B.41, we know that $\varepsilon = \text{inj}_{\times 2}(\varepsilon_1) = \varepsilon_1 \wp \varepsilon_2$, where $\varepsilon_1 = \text{extract}_{\times 2}(\varepsilon)$, $\varepsilon_2 = [(\text{?}_\delta \times \text{?}_\delta, \text{inj}_? \times \text{inj}_?, \text{inj}_\times(\text{inj}_? \times \text{inj}_?))]$ and $\varepsilon_1 : \mathbf{G}_1 \times \mathbf{G}_2 \sim \text{?}_\delta \times \text{?}_\delta$. We know by premise that $\Delta; \Gamma \vdash \pi_1(\varepsilon_1) \mathbf{u}_1 :: \text{?}_\delta : \text{?}_\delta$ and $\Delta; \Gamma \vdash \pi_2(\varepsilon_1) \mathbf{u}_2 :: \text{?}_\delta : \text{?}_\delta$. Then by the induction hypotheses we know that $\Delta; \Gamma \vdash \pi_1(\varepsilon_1) \mathbf{u}_1 :: \text{?}_\delta \preceq \pi_1(\varepsilon_1) \mathbf{u}_1 :: \text{?}_\delta : \text{?}_\delta$ and $\Delta; \Gamma \vdash \pi_2(\varepsilon_1) \mathbf{u}_2 :: \text{?}_\delta \preceq \pi_2(\varepsilon_1) \mathbf{u}_2 :: \text{?}_\delta : \text{?}_\delta$. Then, the result follows directly by Prop B.32 (Compatibility of pairs).

- If $\mathbf{s} = \mathbf{t}'$ then:

$$(\mathbf{G}\text{asc}) \frac{\Delta; \Gamma \vdash \mathbf{t}' : \mathbf{G}' \quad \varepsilon : \mathbf{G}' \sim \mathbf{G}}{\Delta; \Gamma \vdash \varepsilon \mathbf{t}' :: \mathbf{G} : \mathbf{G}}$$

Since $\Delta; \Gamma \vdash \mathbf{t}' : \mathbf{G}'$, by the induction hypotheses we know that $\Delta; \Gamma \vdash \mathbf{t}' \preceq \mathbf{t}' : \mathbf{G}'$. Then, the result follows directly by Prop B.35 (Compatibility of ascriptions).

Case ($\mathbf{G}\text{pair}$). Then $\mathbf{t} = \langle \mathbf{t}_1, \mathbf{t}_2 \rangle$, and therefore:

$$(\mathbf{G}\text{pair}) \frac{\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}_1 \quad \Delta; \Gamma \vdash \mathbf{t}_2 : \mathbf{G}_2}{\Delta; \Gamma \vdash \langle \mathbf{t}_1, \mathbf{t}_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2}$$

where $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$. We have to prove that

$$\Delta; \Gamma \vdash \langle \mathbf{t}_1, \mathbf{t}_2 \rangle \preceq \langle \mathbf{t}_1, \mathbf{t}_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2$$

Since $\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}_1$ and $\Delta; \Gamma \vdash \mathbf{t}_2 : \mathbf{G}_2$, by the induction hypotheses we know that $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_1 : \mathbf{G}_1$ and $\Delta; \Gamma \vdash \mathbf{t}_2 \preceq \mathbf{t}_2 : \mathbf{G}_2$. Then, the result follows directly by Prop B.33 (Compatibility of pairs).

Case (\mathbf{G}_x). Then $\mathbf{t} = x$, and therefore:

$$(\mathbf{G}_x) \frac{x : \mathbf{G} \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x : \mathbf{G}}$$

Then we have to prove that $\Delta; \Gamma \vdash x \preceq x : \mathbf{G}$. The result follows directly by Prop B.34 (Compatibility of variables).

Case (\mathbf{G}_{app}). Then $\mathbf{t} = \mathbf{t}_1 \mathbf{t}_2$, and therefore:

$$(\mathbf{G}_{\text{app}}) \frac{\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}_{11} \longrightarrow \mathbf{G}_{12} \quad \Delta; \Gamma \vdash \mathbf{t}_2 : \mathbf{G}_{11}}{\Delta; \Gamma \vdash \mathbf{t}_1 \mathbf{t}_2 : \mathbf{G}_{12}}$$

where $\mathbf{G} = \mathbf{G}_{12}$. We have to prove that:

$$\Delta; \Gamma \vdash \mathbf{t}_1 \mathbf{t}_2 \preceq \mathbf{t}_1 \mathbf{t}_2 : \mathbf{G}_{12}$$

Since $\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}_{11} \longrightarrow \mathbf{G}_{12}$ and $\Delta; \Gamma \vdash \mathbf{t}_2 : \mathbf{G}_{11}$, by the induction hypothesis we obtain that $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_1 : \mathbf{G}_{11} \longrightarrow \mathbf{G}_{12}$ and $\Delta; \Gamma \vdash \mathbf{t}_2 \preceq \mathbf{t}_2 : \mathbf{G}_{11}$. Then, the result follows directly by Prop B.3 (Compatibility of term application).

Case (\mathbf{G}_{appG}). Then $\mathbf{t} = \mathbf{t}' [\mathbf{F}]$, and therefore:

$$(\mathbf{G}_{\text{appG}}) \frac{\Delta; \Gamma \vdash \mathbf{t}' : \forall \mathbf{X}. \mathbf{G}'' \quad \Delta \vdash \mathbf{F}}{\Delta; \Gamma \vdash \mathbf{t}' [\mathbf{F}] : \mathbf{G}''[\mathbf{F}/\mathbf{X}]}$$

where $\mathbf{G} = \mathbf{G}''[\mathbf{F}/\mathbf{X}]$. Then we have to prove that

$$\Delta; \Gamma \vdash \mathbf{t}' [\mathbf{F}] \preceq \mathbf{t}' [\mathbf{F}] : \mathbf{G}''[\mathbf{F}/\mathbf{X}]$$

Since $\Delta; \Gamma \vdash \mathbf{t}' : \forall \mathbf{X}. \mathbf{G}''$, by the induction hypotheses we know that:

$$\Delta; \Gamma \vdash \mathbf{t}' \preceq \mathbf{t}' : \forall \mathbf{X}. \mathbf{G}''$$

Then the result follows directly by Prop B.37 (Compatibility of type application).

Case ($\mathbf{G}_{\text{pairi}}$). Then $\mathbf{t} = \pi_i(\mathbf{t}')$, and therefore:

$$(\mathbf{G}_{\text{pairi}}) \frac{\Delta; \Gamma \vdash \mathbf{t}' : \mathbf{G}_1 \times \mathbf{G}_2}{\Delta; \Gamma \vdash \pi_i(\mathbf{t}') : \mathbf{G}_i}$$

where $\mathbf{G} = \mathbf{G}_i$. Then we have to prove that

$$\Delta; \Gamma \vdash \pi_i(\mathbf{t}') \preceq \pi_i(\mathbf{t}') : \mathbf{G}_i$$

Since $\Delta; \Gamma \vdash \mathbf{t}' : \mathbf{G}_1 \times \mathbf{G}_2$, by the induction hypothesis we obtain that $\Delta; \Gamma \vdash \mathbf{t}' \preceq \mathbf{t}' : \mathbf{G}_1 \times \mathbf{G}_2$. Then the result follows directly by Prop B.36 (Compatibility of pair projection).

□

Lemma B.28 (Compatibility- \mathbf{G}_b) *If $\theta(\mathbf{b}) = \mathbf{B}$, $\varepsilon : \mathbf{B} \sim \mathbf{G}$ and $\Delta \vdash \Gamma$, then $\Delta; \Gamma \vdash \varepsilon \mathbf{b} :: \mathbf{G} \preceq \varepsilon \mathbf{b} :: \mathbf{G} : \mathbf{G}$.*

PROOF. We know that $\theta(\mathbf{b}) = \mathbf{B}$, $\varepsilon : \mathbf{B} \sim \mathbf{G}$ and $\Delta \vdash \Gamma$. We are required to prove that

$$\Delta; \Gamma \vdash \varepsilon \mathbf{b} :: \mathbf{G} \preceq \varepsilon \mathbf{b} :: \mathbf{G} : \mathbf{G}$$

Let suppose that $\mathbf{t} = \varepsilon \mathbf{b} :: \mathbf{G}$. First, we have to prove that $\Delta; \Gamma \vdash \mathbf{t} : \mathbf{G}$, which follows immediately by the typing rules (**Gb**) and (**Gasc**). Second, we have to prove that for all n, ρ and γ such that $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$, we get that $(n, \rho_1(\gamma_1(\mathbf{t})), \rho_2(\gamma_2(\mathbf{t}))) \in \mathcal{T}_\rho[\mathbf{G}]$. Thus, we are required to prove that $\vdash \rho_1(\gamma_1(\mathbf{t})) : \rho_1(\mathbf{G})$ and $\vdash \rho_2(\gamma_2(\mathbf{t})) : \rho_2(\mathbf{G})$, which follow by Lemma B.60. Since $\rho_i(\gamma_i(\mathbf{t})) = \rho_i(\varepsilon \mathbf{b} :: \rho_i(\mathbf{G}))$ are already values, then we have to prove that: $(n, \rho_1(\gamma_1(\mathbf{t})), \rho_2(\gamma_2(\mathbf{t}))) \in \mathcal{V}_\rho[\mathbf{G}]$. We proceed by case analysis on \mathbf{G} . Since $\varepsilon : \mathbf{B} \sim \mathbf{G}$, then either $\mathbf{G} = \mathbf{B}$ or $\mathbf{G} = ?_\delta$.

- $\mathbf{G} = \mathbf{B}$. Then, we know that $\varepsilon = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\} = \text{reflEv}(\mathbf{B}) = \rho_i(\varepsilon)$, by Lemma B.41. Thus, we are required to prove that $(n, \text{reflEv}(\mathbf{B}) \mathbf{b} :: \mathbf{B}, \text{reflEv}(\mathbf{B}) \mathbf{b} :: \mathbf{B}) \in \mathcal{V}_\rho[\mathbf{B}]$, which follows immediately.
- $\mathbf{G} = ?_\delta$. Let suppose that $\rho_i(\gamma_i(\varepsilon \mathbf{b} :: ?_\delta)) = \rho_i(\varepsilon \mathbf{b} :: \rho_i(?_\delta)) = \rho_i(\varepsilon) \mathbf{u}_i :: \rho_i(?_\delta)$ and $\mathbf{G}^* = \mathbf{B}$. We are required to prove that

$$(n, \rho_1(\varepsilon) \mathbf{u}_1 :: \rho_1(?_\delta), \rho_2(\varepsilon) \mathbf{u}_2 :: \rho_2(?_\delta)) \in \mathcal{V}_\rho[?_\delta]$$

Or what is the same, we are required to prove that for all \mathbf{G}_R and $\varepsilon' (\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R)$, we get that

$$(n, \rho_1(\varepsilon')(\rho_1(\varepsilon) \mathbf{u}_1 :: \rho_1(?_\delta)) :: \rho_1(\mathbf{G}'_R), \rho_2(\varepsilon')(\rho_2(\varepsilon) \mathbf{u}_2 :: \rho_2(?_\delta)) :: \rho_2(\mathbf{G}'_R)) \in \mathcal{T}_\rho[\mathbf{G}'_R]$$

By Lemma B.51, we know that $\rho_i(\varepsilon) \mathbin{\text{\$}} \rho_i(\varepsilon')$ and $\rho_i(\varepsilon \mathbin{\text{\$}} \varepsilon')$ behaves equivalent, where $(\varepsilon \mathbin{\text{\$}} \varepsilon') : \mathbf{G}^* \sim \mathbf{G}'_R$ ($\text{dom}(\rho) = \Delta$). Thus if $\rho_i(\varepsilon \mathbin{\text{\$}} \varepsilon')$ fail, or what is the same if $(\varepsilon \mathbin{\text{\$}} \varepsilon')$ fails, then the result follows immediately because both term behaves equivalent. If $(\varepsilon \mathbin{\text{\$}} \varepsilon')$ does not fail then, we know that $\rho_i(\varepsilon) \mathbin{\text{\$}} \rho_i(\varepsilon') = \rho_i(\varepsilon \mathbin{\text{\$}} \varepsilon')$. Since $\varepsilon : \mathbf{G}^* \sim ?_\delta$, $\varepsilon' : ?_\delta \sim \mathbf{G}'_R$ and $\varepsilon \mathbin{\text{\$}} \varepsilon'$ is defined, by Lemma B.65, we know that $\mathcal{S}(\mathbf{G}^*, ?_\delta) = \mathbf{G}'_R$, therefore, $\mathbf{G}'_R = \mathbf{B}$. Thus, we proceed in the same way as the first case, with evidence $(\varepsilon \mathbin{\text{\$}} \varepsilon') : \mathbf{B} \sim \mathbf{B}$.

□

Lemma B.29 (Compatibility-**Gλ**) *If $\Delta; \Gamma, \mathbf{x} : \mathbf{G}_1 \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}_2$, $\varepsilon : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \sim \mathbf{G}$, then $\Delta; \Gamma \vdash \varepsilon(\lambda \mathbf{x} : \mathbf{G}_1. \mathbf{t}_1) :: \mathbf{G} \preceq \varepsilon(\lambda \mathbf{x} : \mathbf{G}_1. \mathbf{t}_2) :: \mathbf{G} : \mathbf{G}$.*

PROOF. We know that $\Delta; \Gamma, \mathbf{x} : \mathbf{G}_1 \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}_2$, $\varepsilon : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \sim \mathbf{G}$. We are required to prove that $\Delta; \Gamma \vdash \varepsilon(\lambda \mathbf{x} : \mathbf{G}_1. \mathbf{t}_1) :: \mathbf{G} \preceq \varepsilon(\lambda \mathbf{x} : \mathbf{G}_1. \mathbf{t}_2) :: \mathbf{G} : \mathbf{G}$.

Let suppose that $\mathbf{t}_{i\lambda} = \varepsilon(\lambda \mathbf{x} : \mathbf{G}_1. \mathbf{t}_i) :: \mathbf{G}$. First, we have to prove that $\Delta; \Gamma \vdash \mathbf{t}_{i\lambda} : \mathbf{G}$, which follows immediately by the typing rules (**Gλ**) and (**Gasc**). Second, we have to prove that for all n, ρ and γ such that $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$, we get that $(n, \rho_1(\gamma_1(\mathbf{t}_{1\lambda})), \rho_2(\gamma_2(\mathbf{t}_{2\lambda}))) \in \mathcal{T}_\rho[\mathbf{G}]$. Thus, we are required to prove that $\vdash \rho_1(\gamma_1(\mathbf{t}_{1\lambda})) : \rho_1(\mathbf{G})$ and $\vdash \rho_2(\gamma_2(\mathbf{t}_{2\lambda})) : \rho_2(\mathbf{G})$, which follow by Lemma B.60. Since $\rho_i(\gamma_i(\mathbf{t}_{i\lambda})) = \rho_i(\varepsilon)(\lambda \mathbf{x} : \rho_i(\mathbf{G}_1). \rho_i(\gamma_i(\mathbf{t}_i))) :: \rho_i(\mathbf{G})$ are already values, then we have to prove that: $(n, \rho_1(\gamma_1(\mathbf{t}_{1\lambda})), \rho_2(\gamma_2(\mathbf{t}_{2\lambda}))) \in \mathcal{V}_\rho[\mathbf{G}]$. We proceed by case analysis on \mathbf{G} . Since $\varepsilon : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \sim \mathbf{G}$, then either $\mathbf{G} = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$, for some \mathbf{G}'_1 and \mathbf{G}'_2 , or $\mathbf{G} = ?_\delta$.

- $\mathbf{G} = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$. We are required to prove that for all $n' \leq n$, $\mathbf{v}'_1 = \varepsilon'_1 \mathbf{u}'_1 :: \rho_1(\mathbf{G}'_1)$, $\mathbf{v}'_2 = \varepsilon'_2 \mathbf{u}'_2 :: \rho_2(\mathbf{G}'_2)$. $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[\mathbf{G}'_1]$, we get that

$$(n', \rho_1(\gamma_1(\mathbf{t}_{1\lambda})) \mathbf{v}'_1, \rho_2(\gamma_2(\mathbf{t}_{2\lambda})) \mathbf{v}'_2) \in \mathcal{F}_\rho[\mathbf{G}'_2]$$

Note that $\text{dom}(\rho_i(\varepsilon)) = \rho_i(\text{dom}(\varepsilon))$ and $\text{dom}(\varepsilon) : \mathbf{G}'_1 \sim \mathbf{G}_1$. Since $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[\mathbf{G}'_1]$, $\text{dom}(\varepsilon) : \mathbf{G}'_1 \sim \mathbf{G}_1$ and $(n', \rho) \in \mathcal{D}[\Delta]$, by the Lemma 6.10, we get that

$$\triangleright (n', \rho_1(\text{dom}(\varepsilon)) \mathbf{v}'_1 :: \rho_1(\mathbf{G}_1), \rho_2(\text{dom}(\varepsilon)) \mathbf{v}'_2 :: \rho_2(\mathbf{G}_1)) \in \mathcal{F}_\rho[\mathbf{G}_1]$$

Then, we know that both term behaves equivalent. If both terms fail the result follows immediately. Otherwise, if $(\varepsilon'_1 \mathbin{\text{\$}} \text{dom}(\rho_1(\varepsilon)))$ is defined, we know by the operational semantics that

$$\rho_1(\gamma_1(\mathbf{t}_{1\lambda})) \mathbf{v}'_1 \xrightarrow{1} \text{cod}(\rho_1(\varepsilon)) (\rho_1(\gamma_1(\mathbf{t}_1)) [(\varepsilon'_1 \mathbin{\text{\$}} \text{dom}(\rho_1(\varepsilon))) \mathbf{u}'_1 :: \rho_1(\mathbf{G}_1)] / \mathbf{x}) :: \rho_1(\mathbf{G}'_2)$$

$$\rho_2(\gamma_2(\mathbf{t}_{2\lambda})) \mathbf{v}'_2 \xrightarrow{*} \text{cod}(\rho_2(\varepsilon)) (\rho_2(\gamma_2(\mathbf{t}_2)) [(\varepsilon'_2 \mathbin{\text{\$}} \text{dom}(\rho_2(\varepsilon))) \mathbf{u}'_2 :: \rho_2(\mathbf{G}_1)] / \mathbf{x}) :: \rho_2(\mathbf{G}'_2)$$

Since $(\varepsilon'_1 \mathbin{\text{\$}} \text{dom}(\rho_1(\varepsilon)))$ is defined, we get that

$$\triangleright (n', (\varepsilon'_1 \mathbin{\text{\$}} \rho_1(\text{dom}(\varepsilon))) \mathbf{u}'_1 :: \rho_1(\mathbf{G}_1), (\varepsilon'_2 \mathbin{\text{\$}} \rho_2(\text{dom}(\varepsilon))) \mathbf{u}'_2 :: \rho_2(\mathbf{G}_1)) \in \mathcal{V}_\rho[\mathbf{G}_1]$$

We instantiate the premise $\Delta; \Gamma, \mathbf{x} : \mathbf{G}_1 \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}_2$, with n', ρ and $\gamma' = \gamma[\mathbf{x} : \rho(\mathbf{G}_1)] \mapsto (\mathbf{v}''_1, \mathbf{v}''_2)$, where $\mathbf{v}''_i = (\varepsilon'_i \mathbin{\text{\$}} \rho_i(\text{dom}(\varepsilon))) \mathbf{u}'_i :: \rho_i(\mathbf{G}_1)$ and $\triangleright (n', \mathbf{v}''_1, \mathbf{v}''_2) \in \mathcal{V}_\rho[\mathbf{G}_1]$. Note that $(n', \rho) \in \mathcal{D}[\Delta]$ by the definition of $\mathcal{D}[\Delta]$ and $(n', \gamma') \in \mathcal{G}_\rho[\Gamma, \mathbf{x} : \rho(\mathbf{G}_1)]$ because $(n', \gamma) \in \mathcal{G}_\rho[\Gamma]$ and $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[\mathbf{G}_1]$. Then, we know that

$$\triangleright (n', \rho_1(\gamma_1(\mathbf{t}_1)) [\mathbf{v}''_1 / \mathbf{x}], \rho_2(\gamma_2(\mathbf{t}_2)) [\mathbf{v}''_2 / \mathbf{x}]) \in \mathcal{F}_\rho[\mathbf{G}_2]$$

Note that $\rho_i(\gamma'_i(\mathbf{t}_i)) = \rho_i(\gamma_i(\mathbf{t}_i)) [\mathbf{v}''_i / \mathbf{x}]$. If the above terms reduce to error then the result follows immediately. Otherwise, we know that for all $j < n'$, if $\rho_1(\gamma_1(\mathbf{t}_1)) [\mathbf{v}''_1 / \mathbf{x}] \mapsto^j \mathbf{v}_{1s}$ then we know that $\rho_2(\gamma_2(\mathbf{t}_2)) [\mathbf{v}''_2 / \mathbf{x}] \mapsto^* \mathbf{v}_{2s}$ and $\triangleright^j (n', \mathbf{v}_{1s}, \mathbf{v}_{2s}) \in \mathcal{V}_\rho[\mathbf{G}_2]$. Since $\triangleright^j (n', \mathbf{v}_{1s}, \mathbf{v}_{2s}) \in \mathcal{V}_\rho[\mathbf{G}_2]$ and $\text{cod}(\varepsilon) : \mathbf{G}_2 \sim \mathbf{G}'_2$, by Lemma 6.10, we get that

$$\triangleright^j (n', \rho_1(\text{cod}(\varepsilon)) \mathbf{v}_{1s} :: \rho_1(\mathbf{G}'_2), \rho_2(\text{cod}(\varepsilon)) \mathbf{v}_{2s} :: \rho_2(\mathbf{G}'_2)) \in \mathcal{F}_\rho[\mathbf{G}'_2]$$

If transitivity fails both term fail and the result follows immediately. Otherwise, let us suppose that $\mathbf{v}_{is} = \varepsilon_{is} \mathbf{u}_{is} :: \rho_i(\mathbf{G}_2)$, then we know that

$$\triangleright^j (n', (\varepsilon_{1s} \mathbin{\text{\$}} \rho_1(\text{cod}(\varepsilon))) \mathbf{u}_{1s} :: \rho_1(\mathbf{G}'_2), (\varepsilon_{2s} \mathbin{\text{\$}} \rho_2(\text{cod}(\varepsilon))) \mathbf{u}_{2s} :: \rho_2(\mathbf{G}'_2)) \in \mathcal{V}_\rho[\mathbf{G}'_2]$$

Let us suppose that $\mathbf{v}_{if} = (\varepsilon_{is} \mathbin{\text{\$}} \rho_i(\text{cod}(\varepsilon))) \mathbf{u}_{is} :: \rho_i(\mathbf{G}'_2)$, then we get that $\triangleright^j (n', \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\mathbf{G}'_2]$. Therefore, we know that

$$\begin{aligned} \rho_1(\gamma_1(\mathbf{t}_{1\lambda})) \mathbf{v}'_1 &\xrightarrow{1} \text{cod}(\rho_1(\varepsilon)) (\rho_1(\gamma_1(\mathbf{t}_1)) [(\varepsilon'_1 \mathbin{\text{\$}} \text{dom}(\rho_1(\varepsilon))) \mathbf{u}'_1 :: \rho_1(\mathbf{G}_1)] / \mathbf{x}) :: \rho_1(\mathbf{G}'_2) \xrightarrow{j} \\ &\rho_1(\text{cod}(\varepsilon)) \mathbf{v}_{1s} :: \rho_1(\mathbf{G}'_2) \xrightarrow{0} \mathbf{v}_{1f} \\ \rho_2(\gamma_2(\mathbf{t}_{2\lambda})) \mathbf{v}'_2 &\xrightarrow{*} \text{cod}(\rho_2(\varepsilon)) (\rho_2(\gamma_2(\mathbf{t}_2)) [(\varepsilon'_2 \mathbin{\text{\$}} \text{dom}(\rho_2(\varepsilon))) \mathbf{u}'_2 :: \rho_2(\mathbf{G}_1)] / \mathbf{x}) :: \rho_2(\mathbf{G}'_2) \xrightarrow{*} \\ &\rho_2(\text{cod}(\varepsilon)) \mathbf{v}_{2s} :: \rho_2(\mathbf{G}'_2) \xrightarrow{*} \mathbf{v}_{2f} \end{aligned}$$

Thus, we are required to prove that $\triangleright^{1+j} (n', \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\mathbf{G}'_2]$, but we already know that $\triangleright^j (n', \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\mathbf{G}'_2]$, therefore, by Lemma B.45, the result follows immediately.

- $\mathbf{G} = ?_\delta$. Let suppose that $\rho_i(\gamma_i(\mathbf{t}_{i\Lambda})) = \rho_i(\varepsilon)(\lambda\mathbf{x} : \rho_i(\mathbf{G}_1). \rho_i(\gamma_i(\mathbf{t}_i))) :: \rho_i(\mathbf{G}) = \rho_i(\varepsilon) \mathbf{u}_i :: \rho_i(?_\delta)$ and $\mathbf{G}^* = \mathbf{G}_1 \rightarrow \mathbf{G}_2$. We are required to prove that

$$(n, \rho_1(\varepsilon) \mathbf{u}_1 :: \rho_1(?_\delta), \rho_2(\varepsilon) \mathbf{u}_2 :: \rho_2(?_\delta)) \in \mathcal{V}_\rho[?_\delta]$$

Or what is the same, we are required to prove that for all \mathbf{G}_R and $\vdash \varepsilon : \delta \rightarrow \mathbf{G}_R$, we get that

$$(n, \rho_1(\varepsilon')(\rho_1(\varepsilon) \mathbf{u}_1 :: \rho_1(?_\delta)) :: \rho_1(\mathbf{G}_R), \rho_2(\varepsilon')(\rho_2(\varepsilon) \mathbf{u}_2 :: \rho_2(?_\delta)) :: \rho_2(\mathbf{G}_R)) \in \mathcal{T}_\rho[\mathbf{G}_R]$$

By Lemma B.51, we know that $\rho_i(\varepsilon) \mathbin{\varepsilon} \rho_i(\varepsilon')$ and $\rho_i(\varepsilon \mathbin{\varepsilon} \varepsilon')$ behaves equivalent, where $(\varepsilon \mathbin{\varepsilon} \varepsilon') : \mathbf{G}^* \sim \mathbf{G}_R$. Thus if $\rho_i(\varepsilon \mathbin{\varepsilon} \varepsilon')$ fail, or what is the same if $(\varepsilon \mathbin{\varepsilon} \varepsilon')$ fails, then the result follows immediately because both term behaves equivalent. If $(\varepsilon \mathbin{\varepsilon} \varepsilon')$ does not fail then, we know that $\rho_i(\varepsilon) \mathbin{\varepsilon} \rho_i(\varepsilon') = \rho_i(\varepsilon \mathbin{\varepsilon} \varepsilon')$. Since $\varepsilon : \mathbf{G}^* \sim ?_\delta$, $\varepsilon' : ?_\delta \sim \mathbf{G}_R$ and $\varepsilon \mathbin{\varepsilon} \varepsilon'$ is defined, by Lemma B.65, we know that $\mathcal{S}(\mathbf{G}^*, ?_\delta) = \mathbf{G}_R$, therefore, $\mathbf{G}_R = ?_\delta \rightarrow ?_\delta$. Thus, we proceed in the same way as the first case, with evidence $(\varepsilon \mathbin{\varepsilon} \varepsilon') : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \sim ?_\delta \rightarrow ?_\delta$.

□

Lemma B.30 (Compatibility- $\mathbf{G}\Lambda$) *If $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}'$, $\varepsilon : \forall \mathbf{X}. \mathbf{G}' \sim \mathbf{G}$ and $\Delta \vdash \Gamma$, then $\Delta; \Gamma \vdash \varepsilon(\Lambda\mathbf{X}. \mathbf{t}_1) :: \mathbf{G} \preceq \varepsilon(\Lambda\mathbf{X}. \mathbf{t}_2) :: \mathbf{G} : \mathbf{G}$.*

PROOF. We know that $\Delta, \mathbf{X} \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}'$, $\varepsilon : \forall \mathbf{X}. \mathbf{G}' \sim \mathbf{G}$ and $\Delta \vdash \Gamma$. We are required to prove that $\Delta; \Gamma \vdash \varepsilon(\Lambda\mathbf{X}. \mathbf{t}_1) :: \mathbf{G} \preceq \varepsilon(\Lambda\mathbf{X}. \mathbf{t}_2) :: \mathbf{G} : \mathbf{G}$. Let suppose that $\mathbf{t}_{i\Lambda} = \varepsilon(\Lambda\mathbf{X}. \mathbf{t}_i) :: \mathbf{G}$. First, we have to prove that $\Delta; \Gamma \vdash \mathbf{t}_{i\Lambda} : \mathbf{G}$, which follows immediately by the typing rules ($\mathbf{G}\Lambda$) and ($\mathbf{G}\text{asc}$). Second, we have to prove that for all n, ρ and γ such that $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$, we get that $(n, \rho_1(\gamma_1(\mathbf{t}_{1\Lambda})), \rho_2(\gamma_2(\mathbf{t}_{2\Lambda}))) \in \mathcal{T}_\rho[\mathbf{G}]$. Thus, we are required to prove that $\vdash \rho_1(\gamma_1(\mathbf{t}_{1\Lambda})) : \rho_1(\mathbf{G})$ and $\vdash \rho_2(\gamma_2(\mathbf{t}_{2\Lambda})) : \rho_2(\mathbf{G})$, which follow by Lemma B.60. Since $\rho_i(\gamma_i(\mathbf{t}_{i\Lambda})) = \rho_i(\varepsilon)(\Lambda\mathbf{X}. \rho_i(\gamma_i(\mathbf{t}_i))) :: \rho_i(\mathbf{G})$ are already values, then we have to prove that $(n, \rho_1(\gamma_1(\mathbf{t}_{1\Lambda})), \rho_2(\gamma_2(\mathbf{t}_{2\Lambda}))) \in \mathcal{V}_\rho[\mathbf{G}]$. We proceed by case analysis on \mathbf{G} . Since $\varepsilon : \forall \mathbf{X}. \mathbf{G}' \sim \mathbf{G}$, then either $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}''$, for some \mathbf{G}'' , or $\mathbf{G} = ?_\delta$.

- $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}''$. We are required to prove that for all $\vdash \mathbf{F}_1, \vdash \mathbf{F}_2$ and $R \in \text{REL}[\mathbf{F}_1, \mathbf{F}_2]$, we get that

$$(n, \rho_1(\gamma_1(\mathbf{t}_{1\Lambda})) [\mathbf{F}_1], \rho_2(\gamma_2(\mathbf{t}_{2\Lambda})) [\mathbf{F}_2]) \in \mathcal{T}_{\rho; \mathbf{X} \mapsto (\mathbf{F}_1, \mathbf{F}_2, R)}[\mathbf{G}'']$$

Let suppose that $\rho' = \rho; \mathbf{X} \mapsto (\mathbf{F}_1, \mathbf{F}_2, R)$. We know by the operational semantics that

$$\rho_1(\gamma_1(\mathbf{t}_{1\Lambda})) [\mathbf{F}_1] \longrightarrow^1 \text{schm}(\rho_1(\varepsilon))[\mathbf{X} \mapsto \mathbf{F}_1] (\rho_1(\gamma_1(\mathbf{t}_1))[\mathbf{X} \mapsto \mathbf{F}_1]) :: \rho_1(\mathbf{G}'')[\mathbf{F}_1/\mathbf{X}]$$

$$\rho_2(\gamma_2(\mathbf{t}_{2\Lambda})) [\mathbf{F}_2] \longrightarrow^1 \text{schm}(\rho_2(\varepsilon))[\mathbf{X} \mapsto \mathbf{F}_2] (\rho_2(\gamma_2(\mathbf{t}_2))[\mathbf{X} \mapsto \mathbf{F}_2]) :: \rho_2(\mathbf{G}'')[\mathbf{F}_2/\mathbf{X}]$$

Note that $\text{schm}(\rho_i(\varepsilon))[\mathbf{X} \mapsto \mathbf{F}_i] = \rho_i(\text{schm}(\varepsilon))[\mathbf{X} \mapsto \mathbf{F}_i] = \rho'_i(\text{schm}(\varepsilon))$ and $\text{schm}(\varepsilon) : \mathbf{G}' \sim \mathbf{G}''$. We instantiate the premise $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}'$, with $n-1, \rho', \gamma$. Note that $(n-1, \gamma) \in \mathcal{G}_{\rho'}[\Gamma]$ by the definition of $\mathcal{G}_\rho[\Gamma]$ and $(n-1, \rho') \in \mathcal{D}[\Delta, \mathbf{X}]$ because $(n-1, \rho) \in \mathcal{D}[\Delta]$ and $R \in \text{REL}[\mathbf{F}_1, \mathbf{F}_2]$. Then, we know that

$$\triangleright (n, \rho'_1(\gamma_1(\mathbf{t}_1)), \rho'_2(\gamma_2(\mathbf{t}_2))) \in \mathcal{T}_{\rho'}[\mathbf{G}']$$

Note that $\rho'_i(\gamma_i(\mathbf{t}_i)) = \rho_i(\gamma_i(\mathbf{t}_i))[\mathbf{X} \mapsto \mathbf{F}_i]$. If the above terms reduce to error then the result follows immediately. Otherwise, we know that for all $j < n-1$, if $\rho_1(\gamma_1(\mathbf{t}_1))[\mathbf{X} \mapsto F[1]] \mapsto^j \mathbf{v}_{1s}$ then we know that $\rho_2(\gamma_2(\mathbf{t}_2))[\mathbf{X} \mapsto F[2]] \mapsto^* \mathbf{v}_{2s}$ and $\triangleright^{1+j} (n, \mathbf{v}_{1s}, \mathbf{v}_{2s}) \in$

$\mathcal{V}_{\rho'}[\mathbf{G}']$. Since $\triangleright^{1+j} (n, \mathbf{v}_{1s}, \mathbf{v}_{2s}) \in \mathcal{V}_{\rho'}[\mathbf{G}]$, $\text{schm}(\varepsilon) : \mathbf{G}' \sim \mathbf{G}''$ and $(n-1, \rho') \in \mathcal{D}[\Delta, \mathbf{X}]$ by Lemma 6.10, we get that

$$\triangleright^{1+j} (n, \rho'_1(\text{schm}(\varepsilon)) \mathbf{v}_{1s} :: \rho'_1(\mathbf{G}''), \rho'_2(\text{schm}(\varepsilon)) \mathbf{v}_{2s} :: \rho'_2(\mathbf{G}'')) \in \mathcal{J}_{\rho'}[\mathbf{G}'']$$

If transitivity fails both term fail and the result follows immediately. Otherwise, let us suppose that $v_{is} = \varepsilon_{is} \mathbf{u}_{is} :: \rho'_i(\mathbf{G}')$, then we know that

$$\triangleright^{1+j} (n, (\varepsilon_{1s} \circ \rho'_1(\text{schm}(\varepsilon))) \mathbf{u}_{1s} :: \rho'_1(\mathbf{G}''), (\varepsilon_{2s} \circ \rho'_2(\text{schm}(\varepsilon))) \mathbf{u}_{2s} :: \rho'_2(\mathbf{G}'')) \in \mathcal{V}_{\rho'}[\mathbf{G}'']$$

Let us suppose that $\mathbf{v}_{if} = (\varepsilon_{is} \circ \rho'_i(\text{schm}(\varepsilon))) \mathbf{u}_{is} :: \rho'_i(\mathbf{G}')$, then we get that $\triangleright^{1+j} (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho'}[\mathbf{G}'']$. Therefore, we know that

$$\begin{aligned} \rho_1(\gamma_1(\mathbf{t}_{1\Lambda})) [\mathbf{F}_1] &\longrightarrow^1 \text{schm}(\rho_1(\varepsilon))[\mathbf{X} \mapsto \mathbf{F}_1] (\rho_1(\gamma_1(\mathbf{t}_1))[\mathbf{X} \mapsto \mathbf{F}_1] :: \rho_1(\mathbf{G}'')[\mathbf{F}_1/\mathbf{X}]) \longrightarrow^j \\ &\quad \rho_1(\text{schm}(\varepsilon))[\mathbf{X} \mapsto \mathbf{F}_1] \mathbf{v}_{1s} :: \rho_1(\mathbf{G}'')[\mathbf{F}_1/\mathbf{X}] \longrightarrow^0 \mathbf{v}_{1f} \\ \rho_2(\gamma_2(\mathbf{t}_{2\Lambda})) [\mathbf{F}_2] &\longrightarrow^* \text{schm}(\rho_2(\varepsilon))[\mathbf{X} \mapsto \mathbf{F}_2] (\rho_2(\gamma_2(\mathbf{t}_2))[\mathbf{X} \mapsto \mathbf{F}_2] :: \rho_2(\mathbf{G}'')[\mathbf{F}_2/\mathbf{X}]) \longrightarrow^* \\ &\quad \rho_2(\text{schm}(\varepsilon))[\mathbf{X} \mapsto \mathbf{F}_2] \mathbf{v}_{2s} :: \rho_2(\mathbf{G}'')[\mathbf{F}_2/\mathbf{X}] \longrightarrow^* \mathbf{v}_{2f} \end{aligned}$$

Thus, we are required to prove that $\triangleright^{1+j} (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho'}[\mathbf{G}'']$ which already we proved. Therefore, the result follows immediately.

- $\mathbf{G} = ?_\delta$. Let suppose that $\rho_i(\gamma_i(\mathbf{t}_{i\Lambda})) = \rho_i(\varepsilon) (\Delta \mathbf{X}. \rho_i(\gamma_i(\mathbf{t}_i))) :: \rho_i(\mathbf{G}) = \rho_i(\varepsilon) \mathbf{u}_i :: \rho_i(?_\delta)$ and $\mathbf{G}^* = \forall \mathbf{X}. \mathbf{G}'$. We are required to prove that

$$(n, \rho_1(\varepsilon) \mathbf{u}_1 :: \rho_1(?_\delta), \rho_2(\varepsilon) \mathbf{u}_2 :: \rho_2(?_\delta)) \in \mathcal{V}_\rho[?_\delta]$$

Or what is the same, we are required to prove that for all \mathbf{G}_R and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$, we get that

$$(n, \rho_1(\varepsilon') (\rho_1(\varepsilon) \mathbf{u}_1 :: \rho_1(?_\delta)) :: \rho_1(\mathbf{G}_R), \rho_2(\varepsilon') (\rho_2(\varepsilon) \mathbf{u}_2 :: \rho_2(?_\delta)) :: \rho_2(\mathbf{G}_R)) \in \mathcal{J}_\rho[\mathbf{G}_R]$$

By Lemma B.51, we know that $\rho_i(\varepsilon) \circ \rho_i(\varepsilon')$ and $\rho_i(\varepsilon \circ \varepsilon')$ behaves equivalent, where $(\varepsilon \circ \varepsilon') : \mathbf{G}^* \sim \mathbf{G}_R$. Thus if $\rho_i(\varepsilon \circ \varepsilon')$ fail, or what is the same if $(\varepsilon \circ \varepsilon')$ fails, then the result follows immediately because both term behaves equivalent. If $(\varepsilon \circ \varepsilon')$ does not fail then, we know that $\rho_i(\varepsilon) \circ \rho_i(\varepsilon') = \rho_i(\varepsilon \circ \varepsilon')$. Since $\varepsilon : \mathbf{G}^* \sim ?_\delta$, $\varepsilon' : ?_\delta \sim \mathbf{G}_R$ and $\varepsilon \circ \varepsilon'$ is defined, by Lemma B.65, we know that $\mathcal{S}(\mathbf{G}^*, ?_\delta) = \mathbf{G}_R$, therefore, $\mathbf{G}_R = \forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}$. Thus, we proceed in the same way as the first case, with evidence $(\varepsilon \circ \varepsilon') : \forall \mathbf{X}. \mathbf{G}' \sim \forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}$.

□

Lemma B.31 (Compatibility-Gpairu) *If $\Delta; \Gamma \vdash \pi_1(\varepsilon) \mathbf{u}_1 :: \mathbf{G}'_1 \preceq \pi_1(\varepsilon) \mathbf{u}'_1 :: \mathbf{G}'_1 : \mathbf{G}'_1$, $\Delta; \Gamma \vdash \pi_2(\varepsilon) \mathbf{u}_2 :: \mathbf{G}'_2 \preceq \pi_2(\varepsilon) \mathbf{u}'_2 :: \mathbf{G}'_2 : \mathbf{G}'_2$, and $\varepsilon : \mathbf{G}_1 \times \mathbf{G}_2 \sim \mathbf{G}'_1 \times \mathbf{G}'_2$, then $\Delta; \Gamma \vdash \varepsilon \langle \mathbf{u}_1, \mathbf{u}_2 \rangle :: \mathbf{G}'_1 \times \mathbf{G}'_2 \preceq \varepsilon \langle \mathbf{u}'_1, \mathbf{u}'_2 \rangle :: \mathbf{G}'_1 \times \mathbf{G}'_2 : \mathbf{G}'_1 \times \mathbf{G}'_2$.*

PROOF. We know that $\Delta; \Gamma \vdash \pi_1(\varepsilon) \mathbf{u}_1 :: \mathbf{G}'_1 \preceq \pi_1(\varepsilon) \mathbf{u}'_1 :: \mathbf{G}'_1 : \mathbf{G}'_1$, $\Delta; \Gamma \vdash \pi_2(\varepsilon) \mathbf{u}_2 :: \mathbf{G}'_2 \preceq \pi_2(\varepsilon) \mathbf{u}'_2 :: \mathbf{G}'_2 : \mathbf{G}'_2$, and $\varepsilon : \mathbf{G}_1 \times \mathbf{G}_2 \sim \mathbf{G}'_1 \times \mathbf{G}'_2$. We are required to prove that $\Delta; \Gamma \vdash \varepsilon \langle \mathbf{u}_1, \mathbf{u}_2 \rangle :: \mathbf{G}'_1 \times \mathbf{G}'_2 \preceq \varepsilon \langle \mathbf{u}'_1, \mathbf{u}'_2 \rangle :: \mathbf{G}'_1 \times \mathbf{G}'_2 : \mathbf{G}'_1 \times \mathbf{G}'_2$. Let suppose that $\mathbf{v}_1 = \varepsilon \langle \mathbf{u}_1, \mathbf{u}_2 \rangle :: \mathbf{G}'_1 \times \mathbf{G}'_2$ and $\mathbf{v}_2 = \varepsilon \langle \mathbf{u}'_1, \mathbf{u}'_2 \rangle :: \mathbf{G}'_1 \times \mathbf{G}'_2$. First, we have to prove that $\Delta; \Gamma \vdash \mathbf{v}_i : \mathbf{G}'_1 \times \mathbf{G}'_2$, which follows immediately by the typing rules (Gpair) and (Gasc). Second, we have to prove that for all n , ρ and γ such that $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$, we get that $(n, \rho_1(\gamma_1(\mathbf{v}_1)), \rho_2(\gamma_2(\mathbf{v}_2))) \in$

$\mathcal{T}_\rho[\mathbf{G}'_1 \times \mathbf{G}'_2]$. Thus, we are required to prove that $\vdash \rho_1(\gamma_1(\mathbf{v}_1)) : \rho_1(\mathbf{G}'_1 \times \mathbf{G}'_2)$ and $\vdash \rho_2(\gamma_2(\mathbf{v}_2)) : \rho_2(\mathbf{G}'_1 \times \mathbf{G}'_2)$, which follow by Lemma B.60. Since $\rho_1(\gamma_1(\mathbf{v}_1)) = \rho_1(\varepsilon) (\langle \rho_1(\gamma_1(\mathbf{u}_1)), \rho_1(\gamma_1(\mathbf{u}_2)) \rangle) :: \rho_1(\mathbf{G}'_1 \times \mathbf{G}'_2)$ and $\rho_2(\gamma_2(\mathbf{v}_2)) = \rho_2(\varepsilon) (\langle \rho_2(\gamma_2(\mathbf{u}'_1)), \rho_2(\gamma_2(\mathbf{u}'_2)) \rangle) :: \rho_2(\mathbf{G}'_1 \times \mathbf{G}'_2)$ are already values, then we have to prove that $(n, \rho_1(\gamma_1(\mathbf{v}_1)), \rho_2(\gamma_2(\mathbf{v}_2))) \in \mathcal{V}_\rho[\mathbf{G}'_1 \times \mathbf{G}'_2]$. Or what is the same, by the definition of $\mathcal{V}_\rho[\mathbf{G}'_1 \times \mathbf{G}'_2]$, we are required to prove that $(n, \pi_1(\rho_1(\gamma_1(\mathbf{v}_1))), \pi_1(\rho_2(\gamma_2(\mathbf{v}_2)))) \in \mathcal{T}_\rho[\mathbf{G}'_1]$ and $(n, \pi_2(\rho_1(\gamma_1(\mathbf{v}_1))), \pi_2(\rho_2(\gamma_2(\mathbf{v}_2)))) \in \mathcal{T}_\rho[\mathbf{G}'_2]$. Therefore, we are required to prove that $\triangleright^1 (n, \rho_1(\gamma_1(\pi_1(\varepsilon) \mathbf{u}_1 :: \mathbf{G}'_1)), \rho_2(\gamma_2(\pi_1(\varepsilon) \mathbf{u}'_1 :: \mathbf{G}'_1))) \in \mathcal{V}_\rho[\mathbf{G}'_1]$ and $\triangleright^1 (n, \rho_1(\gamma_1(\pi_2(\varepsilon) \mathbf{u}_2 :: \mathbf{G}'_2)), \rho_2(\gamma_2(\pi_2(\varepsilon) \mathbf{u}'_2 :: \mathbf{G}'_2))) \in \mathcal{V}_\rho[\mathbf{G}'_2]$, which follows immediately instantiating the premises $\Delta; \Gamma \vdash \pi_1(\varepsilon) \mathbf{u}_1 :: \mathbf{G}'_1 \preceq \pi_1(\varepsilon) \mathbf{u}'_1 :: \mathbf{G}'_1 : \mathbf{G}'_1$ and $\Delta; \Gamma \vdash \pi_2(\varepsilon) \mathbf{u}_2 :: \mathbf{G}'_2 \preceq \pi_2(\varepsilon) \mathbf{u}'_2 :: \mathbf{G}'_2 : \mathbf{G}'_2$, with $n - 1$, ρ and γ . \square

Lemma B.32 (Compatibility-**G**pair $_{\text{?}\delta}$) *If $\Delta; \Gamma \vdash \pi_1(\varepsilon) \mathbf{u}_1 :: \text{?}\delta \preceq \pi_1(\varepsilon) \mathbf{u}'_1 :: \text{?}\delta : \text{?}\delta$, $\Delta; \Gamma \vdash \pi_2(\varepsilon) \mathbf{u}_2 :: \text{?}\delta \preceq \pi_2(\varepsilon) \mathbf{u}'_2 :: \text{?}\delta : \text{?}\delta$, and $\varepsilon : \mathbf{G}_1 \times \mathbf{G}_2 \sim \text{?}\delta \times \text{?}\delta$, then $\Delta; \Gamma \vdash \text{inj}_{\times 2}(\varepsilon) \langle \mathbf{u}_1, \mathbf{u}_2 \rangle :: \text{?}\delta \preceq \text{inj}_{\times 2}(\varepsilon) \langle \mathbf{u}'_1, \mathbf{u}'_2 \rangle :: \text{?}\delta : \text{?}\delta$.*

PROOF. By Lemma B.31 and similar to the function case ascribed to unknown. \square

Lemma B.33 (Compatibility-**G**pair) *If $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}'_1 : \mathbf{G}_1$ and $\Delta; \Gamma \vdash \mathbf{t}_2 \preceq \mathbf{t}'_2 : \mathbf{G}_2$, then $\Delta; \Gamma \vdash \langle \mathbf{t}_1, \mathbf{t}_2 \rangle \preceq \langle \mathbf{t}'_1, \mathbf{t}'_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2$.*

PROOF. We know that $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}'_1 : \mathbf{G}_1$ and $\Delta; \Gamma \vdash \mathbf{t}_2 \preceq \mathbf{t}'_2 : \mathbf{G}_2$. We are required to prove that $\Delta; \Gamma \vdash \langle \mathbf{t}_1, \mathbf{t}_2 \rangle \preceq \langle \mathbf{t}'_1, \mathbf{t}'_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2$. First, we have to prove that $\Delta; \Gamma \vdash \langle \mathbf{t}_1, \mathbf{t}_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2$ and $\Delta; \Gamma \vdash \langle \mathbf{t}'_1, \mathbf{t}'_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2$, which follows immediately by the typing rule (**G**pair). Second, we have to prove that for all n, ρ and γ such that $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$, we get that $(n, \langle \rho_1(\gamma_1(\mathbf{t}_1)), \rho_1(\gamma_1(\mathbf{t}_2)) \rangle, \langle \rho_2(\gamma_2(\mathbf{t}'_1)), \rho_2(\gamma_2(\mathbf{t}'_2)) \rangle) \in \mathcal{T}_\rho[\mathbf{G}_1 \times \mathbf{G}_2]$. Thus, we are required to prove that $\vdash \langle \rho_1(\gamma_1(\mathbf{t}_1)), \rho_1(\gamma_1(\mathbf{t}_2)) \rangle : \rho_1(\mathbf{G}_1) \times \rho_1(\mathbf{G}_2)$ and $\vdash \langle \rho_2(\gamma_2(\mathbf{t}'_1)), \rho_2(\gamma_2(\mathbf{t}'_2)) \rangle : \rho_2(\mathbf{G}_1) \times \rho_2(\mathbf{G}_2)$, which follow by Lemma B.60. Consider arbitrary $i < n$. Either there exists \mathbf{t}_1^* such that $\langle \rho_1(\gamma_1(\mathbf{t}_1)), \rho_1(\gamma_1(\mathbf{t}_2)) \rangle \mapsto^i \mathbf{t}_1^*$ or $\langle \rho_1(\gamma_1(\mathbf{t}_1)), \rho_1(\gamma_1(\mathbf{t}_2)) \rangle \mapsto^i \text{error}$. First, let us suppose that $\langle \rho_1(\gamma_1(\mathbf{t}_1)), \rho_1(\gamma_1(\mathbf{t}_2)) \rangle \mapsto^i \mathbf{t}_1^*$. If \mathbf{t}_1^* is not a value the result holds immediately. Let us suppose $\mathbf{t}_1^* = \mathbf{v}_1$. Hence, by inspection of the operational semantics, it follows that there exist $i_1 < i, i_2 < i - i_1, \mathbf{v}_{11}, \mathbf{v}_{21}$ and \mathbf{v}_1 such that

$$\langle \rho_1(\gamma_1(\mathbf{t}_1)), \rho_1(\gamma_1(\mathbf{t}_2)) \rangle \mapsto^{i_1} \langle \mathbf{v}_{11}, \rho_1(\gamma_1(\mathbf{t}_2)) \rangle \mapsto^{i_2} \langle \mathbf{v}_{11}, \mathbf{v}_{22} \rangle \mapsto^1 \mathbf{v}_1$$

where $\mathbf{v}_1 = \varepsilon_{11} \times \varepsilon_{21} \langle \mathbf{u}_{11}, \mathbf{u}_{21} \rangle :: \rho_1(\mathbf{G}_1) \times \rho_1(\mathbf{G}_2)$, $\mathbf{v}_{11} = \varepsilon_{11} \mathbf{u}_{11} :: \rho_1(\mathbf{G}_1)$ and $\mathbf{v}_{21} = \varepsilon_{21} \mathbf{u}_{21} :: \rho_1(\mathbf{G}_2)$. We instantiate the premise $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}'_1 : \mathbf{G}_1$ with n, ρ and γ to obtain that $(n, \rho_1(\gamma_1(\mathbf{t}_1)), \rho_2(\gamma_2(\mathbf{t}'_1))) \in \mathcal{T}_\rho[\mathbf{G}_1]$. Then, we know that both term behaves equivalent. If both terms fail the result follows immediately. Otherwise, we instantiate $\mathcal{T}_\rho[\mathbf{G}_1]$ with i_1 and \mathbf{v}_{11} (note that $i_1 < i < n$), hence there exists \mathbf{v}_{12} , such that $\rho_2(\gamma_2(\mathbf{t}'_1)) \mapsto^* \mathbf{v}_{12}$ and $\triangleright^{i_1} (n, \mathbf{v}_{11}, \mathbf{v}_{12}) \in \mathcal{V}_\rho[\mathbf{G}_1]$. Next, we instantiate the premise $\Delta; \Gamma \vdash \mathbf{t}_2 \preceq \mathbf{t}'_2 : \mathbf{G}_2$ with $n - i_1, \rho$ and γ to obtain that $\triangleright^{i_1} (n, \rho_1(\gamma_1(\mathbf{t}_2)), \rho_2(\gamma_2(\mathbf{t}'_2))) \in \mathcal{T}_\rho[\mathbf{G}_2]$. Then, we know that both term behaves equivalent. If both terms fail the result follows immediately. Otherwise, we instantiate $\mathcal{T}_\rho[\mathbf{G}_2]$ with i_2 and \mathbf{v}_{21} (note that $i_2 < i - i_1 < n - i_1$), hence there exists \mathbf{v}_{22} , such that $\rho_2(\gamma_2(\mathbf{t}'_2)) \mapsto^* \mathbf{v}_{22}$ and $\triangleright^{i_1+i_2} (n, \mathbf{v}_{21}, \mathbf{v}_{22}) \in \mathcal{V}_\rho[\mathbf{G}_2]$. We know by the operational semantics that

$$\langle \rho_2(\gamma_2(\mathbf{t}'_1)), \rho_2(\gamma_2(\mathbf{t}'_2)) \rangle \mapsto^* \langle \mathbf{v}_{12}, \rho_2(\gamma_2(\mathbf{t}'_2)) \rangle \mapsto^* \langle \mathbf{v}_{12}, \mathbf{v}_{22} \rangle \mapsto^1 \mathbf{v}_2$$

where $\mathbf{v}_2 = \varepsilon_{12} \times \varepsilon_{22} \langle \mathbf{u}_{12}, \mathbf{u}_{22} \rangle :: \rho_2(\mathbf{G}_1) \times \rho_2(\mathbf{G}_2)$, $\mathbf{v}_{12} = \varepsilon_{12} \mathbf{u}_{12} :: \rho_2(\mathbf{G}_1)$ and $\mathbf{v}_{22} = \varepsilon_{22} \mathbf{u}_{22} :: \rho_2(\mathbf{G}_2)$. Also, by the operational semantics we know that $\pi_i(\mathbf{v}_1) \mapsto^1 v_{i1}$ and $\pi_i(\mathbf{v}_2) \mapsto^1 v_{i2}$. We

are required to prove that $\triangleright^{i_1+i_2} (n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}_1 \times \mathbf{G}_2]$. Or what is the same, we are required to prove that $\triangleright^{i_1+i_2} (n, \pi_1(\mathbf{v}_1), \pi_1(\mathbf{v}_2)) \in \mathcal{T}_\rho[\mathbf{G}_1]$ and $\triangleright^{i_1+i_2} (n, \pi_2(\mathbf{v}_1), \pi_2(\mathbf{v}_2)) \in \mathcal{T}_\rho[\mathbf{G}_2]$. Therefore, we are required to prove that $\triangleright^{i_1+i_2+1} (n, \mathbf{v}_{11}, \mathbf{v}_{12}) \in \mathcal{V}_\rho[\mathbf{G}_1]$ $\triangleright^{i_1+i_2+1} (n, \mathbf{v}_{21}, \mathbf{v}_{22}) \in \mathcal{V}_\rho[\mathbf{G}_2]$, which follows by Lemma B.45. Finally, the result holds. \square

Lemma B.34 (Compatibility- \mathbf{Gx}) *If $\mathbf{x} : \mathbf{G} \in \Gamma$ and $\Delta \vdash \Gamma$, then $\Delta; \Gamma \vdash \mathbf{x} \preceq \mathbf{x} : \mathbf{G}$.*

PROOF. We know that $\mathbf{x} : \mathbf{G} \in \Gamma$ and $\Delta \vdash \Gamma$. We are required to prove that $\Delta; \Gamma \vdash \mathbf{x} \preceq \mathbf{x} : \mathbf{G}$. First, we have to prove that $\Delta; \Gamma \vdash \mathbf{x} : \mathbf{G}$, which follows immediately by the typing rule (\mathbf{Gx}). Second, we have to prove that for all n, ρ and γ such that $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$, we get that $(n, \rho_1(\gamma_1(\mathbf{x})), \rho_2(\gamma_2(\mathbf{x}))) \in \mathcal{T}_\rho[\mathbf{G}]$. Thus, we are required to prove that $\vdash \rho_1(\gamma_1(\mathbf{x})) : \rho_1(\mathbf{G})$ and $\vdash \rho_2(\gamma_2(\mathbf{x})) : \rho_2(\mathbf{G})$, which follow by Lemma B.60. Since $\rho_i(\gamma_i(\mathbf{x})) = \gamma_i(\mathbf{x}) = \mathbf{v}_i$ are already values, then we have to prove that $(n, \gamma_1(\mathbf{x}), \gamma_2(\mathbf{x})) \in \mathcal{V}_\rho[\mathbf{G}]$. Or what is the same, we are required to prove that $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$, which follows immediately by definition of $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$. \square

Lemma B.35 (Compatibility- \mathbf{Gasc}) *If $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}'$ and $\varepsilon : \mathbf{G}' \sim \mathbf{G}$, then $\Delta; \Gamma \vdash \varepsilon \mathbf{t}_1 :: \mathbf{G} \preceq \varepsilon \mathbf{t}_2 :: \mathbf{G} : \mathbf{G}$.*

PROOF. We know that $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}'$ and $\varepsilon : \mathbf{G}' \sim \mathbf{G}$. We are required to prove that $\Delta; \Gamma \vdash \varepsilon \mathbf{t}_1 :: \mathbf{G} \preceq \varepsilon \mathbf{t}_2 :: \mathbf{G} : \mathbf{G}$. First, we have to prove that $\Delta; \Gamma \vdash \varepsilon \mathbf{t}_i :: \mathbf{G} : \mathbf{G}$, which follows immediately by the typing rule (\mathbf{Gasc}). Second, we have to prove that for all n, ρ and γ such that $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$, we get that $(n, \rho_1(\varepsilon) \rho_1(\gamma_1(\mathbf{t}_1)) :: \rho_1(\mathbf{G}), \rho_2(\varepsilon) \rho_2(\gamma_2(\mathbf{t}_2)) :: \rho_2(\mathbf{G})) \in \mathcal{T}_\rho[\mathbf{G}]$. Thus, we are required to prove that $\vdash \rho_1(\varepsilon) \rho_1(\gamma_1(\mathbf{t}_1)) :: \rho_1(\mathbf{G}) : \rho_1(\mathbf{G})$ and $\vdash \rho_2(\varepsilon) \rho_2(\gamma_2(\mathbf{t}_2)) :: \rho_2(\mathbf{G}) : \rho_2(\mathbf{G})$, which follow by Lemma B.60. Consider arbitrary $i < n$. Either there exists \mathbf{t}'_1 such that $\rho_1(\varepsilon) \rho_1(\gamma_1(\mathbf{t}_1)) :: \rho_1(\mathbf{G}) \mapsto \mathbf{t}'_1$ or $\rho_1(\varepsilon) \rho_1(\gamma_1(\mathbf{t}_1)) :: \rho_1(\mathbf{G}) \mapsto \mathbf{error}$. First, let us suppose that $\rho_1(\varepsilon) \rho_1(\gamma_1(\mathbf{t}_1)) :: \rho_1(\mathbf{G}) \mapsto \mathbf{t}'_1$. If \mathbf{t}'_1 is not a value the result holds immediately. Let us suppose $\mathbf{t}'_1 = \mathbf{v}'_1$. Hence, by inspection of the operational semantics, it follows that there exist $i_1 < i$, and \mathbf{v}_{11} such that $\rho_1(\varepsilon) \rho_1(\gamma_1(\mathbf{t}_1)) :: \rho_1(\mathbf{G}) \mapsto \mathbf{v}_{11} :: \rho_1(\mathbf{G})$. We instantiate the premise $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}'$ with n, ρ and γ to obtain that $(n, \rho_1(\gamma_1(\mathbf{t}_1)), \rho_2(\gamma_2(\mathbf{t}_2))) \in \mathcal{T}_\rho[\mathbf{G}']$. We instantiate $\mathcal{T}_\rho[\mathbf{G}']$ with i_1 and \mathbf{v}_{11} (note that $i_1 < i < n$), hence there exists \mathbf{v}_{12} , such that $\rho_2(\gamma_2(\mathbf{t}_2)) \mapsto^* \mathbf{v}_{12}$, and $\triangleright^{i_1} (n, \mathbf{v}_{11}, \mathbf{v}_{12}) \in \mathcal{V}_\rho[\mathbf{G}']$. Let us suppose that $v_{1i} = \varepsilon_i \mathbf{u}_i :: \rho_i(\mathbf{G}')$. Since $\triangleright^{i_1} (n, \mathbf{v}_{11}, \mathbf{v}_{12}) \in \mathcal{V}_\rho[\mathbf{G}']$, $\varepsilon : \mathbf{G}' \sim \mathbf{G}$ and $(n - i_1, \rho) \in \mathcal{D}[\Delta]$, by Lemma 6.10, we get that $\triangleright^{i_1} (n, \rho_1(\varepsilon) \mathbf{v}_{11} :: \rho_1(\mathbf{G}), \rho_2(\varepsilon) \mathbf{v}_{12} :: \rho_2(\mathbf{G})) \in \mathcal{T}_\rho[\mathbf{G}]$. Therefore, we get that both term fail and the result hold immediately, or both term reduce and $\triangleright^{i_1} (n, (\varepsilon_1 \mathbin{\text{\textcircled{;}}} \rho_1(\varepsilon)) \mathbf{u}_1 :: \rho_1(\mathbf{G}), (\varepsilon_2 \mathbin{\text{\textcircled{;}}} \rho_2(\varepsilon)) \mathbf{u}_2 :: \rho_2(\mathbf{G})) \in \mathcal{V}_\rho[\mathbf{G}]$. By inspection of the operational semantics, we know that

$$\rho_1(\varepsilon) \rho_1(\gamma_1(\mathbf{t}_1)) :: \rho_1(\mathbf{G}) \mapsto \mathbf{v}_{11} :: \rho_1(\mathbf{G}) \mapsto \mathbf{u}_1 :: \rho_1(\mathbf{G})$$

and

$$\rho_2(\varepsilon) \rho_2(\gamma_2(\mathbf{t}_2)) :: \rho_2(\mathbf{G}) \mapsto^* \rho_2(\varepsilon) \mathbf{v}_{12} :: \rho_2(\mathbf{G}) \mapsto^* (\varepsilon_2 \mathbin{\text{\textcircled{;}}} \rho_2(\varepsilon)) \mathbf{u}_2 :: \rho_2(\mathbf{G})$$

Thus, we are required to prove that $\triangleright^{i_1} (n, (\varepsilon_1 \mathbin{\text{\textcircled{;}}} \rho_1(\varepsilon)) \mathbf{u}_1 :: \rho_1(\mathbf{G}), (\varepsilon_2 \mathbin{\text{\textcircled{;}}} \rho_2(\varepsilon)) \mathbf{u}_2 :: \rho_2(\mathbf{G})) \in \mathcal{V}_\rho[\mathbf{G}]$, which already is proved. Therefore, the result holds. \square

[Compatibility-Eapp] If $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}_1 \longrightarrow \mathbf{G}_2$ and $\Delta; \Gamma \vdash \mathbf{t}'_1 \preceq \mathbf{t}'_2 : \mathbf{G}_1$, then $\Delta; \Gamma \vdash \mathbf{t}_1 \mathbf{t}'_1 \preceq \mathbf{t}_2 \mathbf{t}'_2 : \mathbf{G}_2$.

PROOF. We know that $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}_1 \longrightarrow \mathbf{G}_2$ and $\Delta; \Gamma \vdash \mathbf{t}'_1 \preceq \mathbf{t}'_2 : \mathbf{G}_1$. We are required to prove that $\Delta; \Gamma \vdash \mathbf{t}_1 \mathbf{t}'_1 \preceq \mathbf{t}_2 \mathbf{t}'_2 : \mathbf{G}_2$. First, we have to prove that $\Delta; \Gamma \vdash \mathbf{t}_i \mathbf{t}'_i : \mathbf{G}_2$, which follows immediately by the typing rule (**Gapp**). Second, we have to prove that for all n, ρ and γ such that $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$, we get that $(n, \rho_1(\gamma_1(\mathbf{t}_1)), \rho_1(\gamma_1(\mathbf{t}'_1)), \rho_2(\gamma_2(\mathbf{t}_2)), \rho_2(\gamma_2(\mathbf{t}'_2))) \in \mathcal{J}_\rho[\mathbf{G}_2]$. Thus, we are required to prove that $\vdash \rho_1(\gamma_1(\mathbf{t}_1)) \rho_1(\gamma_1(\mathbf{t}'_1)) : \rho_1(\mathbf{G}_2)$ and $\vdash \rho_2(\gamma_2(\mathbf{t}_2)) \rho_2(\gamma_2(\mathbf{t}'_2)) : \rho_2(\mathbf{G}_2)$, which follow by Lemma B.60. Consider arbitrary $i < n$. Either there exists \mathbf{t}_1^* such that $\rho_1(\gamma_1(\mathbf{t}_1)) \rho_1(\gamma_1(\mathbf{t}'_1)) \mapsto^i \mathbf{t}_1^*$ or $\rho_1(\gamma_1(\mathbf{t}_1)) \rho_1(\gamma_1(\mathbf{t}'_1)) \mapsto^i \mathbf{error}$. First, let us suppose that $\rho_1(\gamma_1(\mathbf{t}_1)) \rho_1(\gamma_1(\mathbf{t}'_1)) \mapsto^i \mathbf{t}_1^*$. If \mathbf{t}_1^* is not a value the result holds immediately. Let us suppose $\mathbf{t}_1^* = \mathbf{v}_1$. Hence, by inspection of the operational semantics, it follows that there exist $i_1 < i, i_2 < i - i_1, i_3 = i - i_1 - i_2, \mathbf{v}_{11}$ and \mathbf{v}_{21} such that $\rho_1(\gamma_1(\mathbf{t}_1)) \rho_1(\gamma_1(\mathbf{t}'_1)) \mapsto^{i_1} \mathbf{v}_{11} \rho_1(\gamma_1(\mathbf{t}'_1)) \mapsto^{i_2} \mathbf{v}_{11} \mathbf{v}_{21} \mapsto^{i_3} \mathbf{v}_1$. We instantiate the premise $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}_1 \rightarrow \mathbf{G}_2$ with n, ρ and γ to obtain that $(n, \rho_1(\gamma_1(\mathbf{t}_1)), \rho_2(\gamma_2(\mathbf{t}_2))) \in \mathcal{J}_\rho[\mathbf{G}_1 \rightarrow \mathbf{G}_2]$. Then, we know that both term behaves equivalent. If both terms fail the result follows immediately. Otherwise, we instantiate $\mathcal{J}_\rho[\mathbf{G}_1 \rightarrow \mathbf{G}_2]$ with i_1 and \mathbf{v}_{11} (note that $i_1 < i < n$), hence there exists \mathbf{v}_{12} , such that $\rho_2(\gamma_2(\mathbf{t}_2)) \mapsto^* \mathbf{v}_{12}$ and $\triangleright^{i_1} (n, \mathbf{v}_{11}, \mathbf{v}_{12}) \in \mathcal{V}_\rho[\mathbf{G}_1 \rightarrow \mathbf{G}_2]$. Next, we instantiate the premise $\Delta; \Gamma \vdash \mathbf{t}'_1 \preceq \mathbf{t}'_2 : \mathbf{G}_1$ with $n - i_1, \rho$ and γ to obtain that $\triangleright^{i_1} (n, \rho_1(\gamma_1(\mathbf{t}'_1)), \rho_2(\gamma_2(\mathbf{t}'_2))) \in \mathcal{J}_\rho[\mathbf{G}_1]$. Then, we know that both term behaves equivalent. If both terms fail the result follows immediately. Otherwise, we instantiate $\mathcal{J}_\rho[\mathbf{G}_1]$ with i_2 and \mathbf{v}_{21} (note that $i_2 < i - i_1 < n - i_1$), hence there exists \mathbf{v}_{22} , such that $\rho_2(\gamma_2(\mathbf{t}'_2)) \mapsto^* \mathbf{v}_{22}$ and $\triangleright^{i_1+i_2} (n, \mathbf{v}_{21}, \mathbf{v}_{22}) \in \mathcal{V}_\rho[\mathbf{G}_1]$. Now, we instantiate $\triangleright^{i_1} (n, \mathbf{v}_{11}, \mathbf{v}_{12}) \in \mathcal{V}_\rho[\mathbf{G}_1 \rightarrow \mathbf{G}_2]$, with $n - i_1 - i_2 \leq n - i_1$ and $\triangleright^{i_1+i_2+1} (n, \mathbf{v}_{21}, \mathbf{v}_{22}) \in \mathcal{V}_\rho[\mathbf{G}_1]$ ($\triangleright^{i_1+i_2} (n, \mathbf{v}_{21}, \mathbf{v}_{22}) \in \mathcal{V}_\rho[\mathbf{G}_1]$ and Lemma B.45), and we get that $\triangleright^{i_1+i_2} (n, \mathbf{v}_{11} \mathbf{v}_{21}, \mathbf{v}_{12} \mathbf{v}_{22}) \in \mathcal{J}_\rho[\mathbf{G}_2]$. Then, we know that both term behaves equivalent. If both terms fail the result follows immediately. Otherwise, we instantiate $\mathcal{J}_\rho[\mathbf{G}_2]$ with i_3 and \mathbf{v}_1 (note that $i_3 = i - i_1 - i_2 < n - i_1 - i_2$), hence there exists \mathbf{v}_2 , such that $\mathbf{v}_{12} \mathbf{v}_{22} \mapsto^* \mathbf{v}_2$ and $\triangleright^{i_1+i_2+i_3} (n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}_2]$. Finally, the result holds. \square

Lemma B.36 (Compatibility-**Gpairi**) *If $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}_1 \times \mathbf{G}_2$ then $\Delta; \Gamma \vdash \pi_i(\mathbf{t}_1) \preceq \pi_i(\mathbf{t}_2) : \mathbf{G}_i$.*

PROOF. We know that $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}_1 \times \mathbf{G}_2$. We are required to prove that $\Delta; \Gamma \vdash \pi_i(\mathbf{t}_1) \preceq \pi_i(\mathbf{t}_2) : \mathbf{G}_i$. First, we have to prove that $\Delta; \Gamma \vdash \pi_i(\mathbf{t}_1) : \mathbf{G}_i$ and $\Delta; \Gamma \vdash \pi_i(\mathbf{t}_2) : \mathbf{G}_i$, which follows immediately by the typing rule (**Gpairi**). Second, we have to prove that for all n, ρ and γ such that $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$, we get that $(n, \pi_i(\rho_1(\gamma_1(\mathbf{t}_1))), \pi_i(\rho_2(\gamma_2(\mathbf{t}_2)))) \in \mathcal{J}_\rho[\mathbf{G}_i]$. Thus, we are required to prove that $\vdash \pi_i(\rho_1(\gamma_1(\mathbf{t}_1))) : \rho_1(\mathbf{G}_i)$ and $\vdash \pi_i(\rho_2(\gamma_2(\mathbf{t}_2))) : \rho_2(\mathbf{G}_i)$, which follow by Lemma B.60. Consider arbitrary $i < n$. Either there exists \mathbf{t}_1^* such that $\pi_i(\rho_1(\gamma_1(\mathbf{t}_1))) \mapsto^i \mathbf{t}_1^*$ or $\pi_i(\rho_1(\gamma_1(\mathbf{t}_1))) \mapsto^i \mathbf{error}$. First, let us suppose that $\pi_i(\rho_1(\gamma_1(\mathbf{t}_1))) \mapsto^i \mathbf{t}_1^*$. If \mathbf{t}_1^* is not a value the result holds immediately. Let us suppose $\mathbf{t}_1^* = \mathbf{v}_1$. Hence, by inspection of the operational semantics, it follows that there exist $i_1 < i, i_2 = i - i_1, \mathbf{v}_{11}$ and \mathbf{v}_1 such that $\pi_i(\rho_1(\gamma_1(\mathbf{t}_1))) \mapsto^{i_1} \pi_i(\mathbf{v}_{11}) \mapsto^{i_2} \mathbf{v}_1$. We instantiate the premise $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \mathbf{G}_1 \times \mathbf{G}_2$ with n, ρ and γ to obtain that $(n, \rho_1(\gamma_1(\mathbf{t}_1)), \rho_2(\gamma_2(\mathbf{t}_2))) \in \mathcal{J}_\rho[\mathbf{G}_1 \times \mathbf{G}_2]$. Then, we know that both term behaves equivalent. If both terms fail the result follows immediately. Otherwise, we instantiate $\mathcal{J}_\rho[\mathbf{G}_1 \times \mathbf{G}_2]$ with i_1 and \mathbf{v}_{11} (note that $i_1 < i < n$), hence there exists \mathbf{v}_{12} , such that $\rho_2(\gamma_2(\mathbf{t}_2)) \mapsto^* \mathbf{v}_{12}$ and $\triangleright^{i_1} (n, \mathbf{v}_{11}, \mathbf{v}_{12}) \in \mathcal{V}_\rho[\mathbf{G}_1 \times \mathbf{G}_2]$. By the definition of $\triangleright^{i_1} (n, \mathbf{v}_{11}, \mathbf{v}_{12}) \in \mathcal{V}_\rho[\mathbf{G}_1 \times \mathbf{G}_2]$, we know that $\triangleright^{i_1} (n, \pi_i(\mathbf{v}_{11}), \pi_i(\mathbf{v}_{12})) \in \mathcal{J}_\rho[\mathbf{G}_i]$. Then, we know that both term behaves equivalent. If both terms fail the result follows immediately. Otherwise, we instantiate

$\mathcal{J}_\rho[\mathbf{G}_i]$ with i_2 and \mathbf{v}_1 (note that $i_2 = i - i_1 < n - i_1$), hence there exists \mathbf{v}_2 , such that $\pi_i(\mathbf{v}_{12}) \xrightarrow{*} \mathbf{v}_2$ and $\triangleright^{i_1+i_2} (n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}_i]$. Thus, the result follows immediately. \square

Lemma B.37 (Compatibility-**G**app**G**) *If $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \forall \mathbf{X}. \mathbf{G}$ and $\Delta \vdash \mathbf{F}$, then $\Delta; \Gamma \vdash \mathbf{t}_1 [\mathbf{F}] \preceq \mathbf{t}_2 [\mathbf{F}] : \mathbf{G}[\mathbf{F}/\mathbf{X}]$.*

PROOF. We know that $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \forall \mathbf{X}. \mathbf{G}$ and $\Delta \vdash \mathbf{F}$. We are required to prove that $\Delta; \Gamma \vdash \mathbf{t}_1 [\mathbf{F}] \preceq \mathbf{t}_2 [\mathbf{F}] : \mathbf{G}[\mathbf{F}/\mathbf{X}]$.

First, we have to prove that $\Delta; \Gamma \vdash \mathbf{t}_i [\mathbf{F}] : \mathbf{G}[\mathbf{F}/\mathbf{X}]$, which follows immediately by the typing rule (**G**app**G**). Second, we have to prove that for all n, ρ and γ such that $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$, we get that $(n, \rho_1(\gamma_1(\mathbf{t}_1)) [\rho_1(\mathbf{F})], \rho_2(\gamma_2(\mathbf{t}_2)) [\rho_2(\mathbf{F})]) \in \mathcal{J}_\rho[\mathbf{G}[\mathbf{F}/\mathbf{X}]]$. Thus, we are required to prove that $\vdash \rho_1(\gamma_1(\mathbf{t}_1)) [\rho_1(\mathbf{F})] : \rho_1(\mathbf{G}[\mathbf{F}/\mathbf{X}])$ and $\vdash \rho_2(\gamma_2(\mathbf{t}_2)) [\rho_2(\mathbf{F})] : \rho_2(\mathbf{G}[\mathbf{F}/\mathbf{X}])$, which follow by Lemma B.60. Consider arbitrary $i < n$. Either there exists \mathbf{t}'_1 such that $\rho_1(\gamma_1(\mathbf{t}_1)) [\rho_1(\mathbf{F})] \mapsto^i \mathbf{t}'_1$ or $\rho_1(\gamma_1(\mathbf{t}_1)) [\rho_1(\mathbf{F})] \mapsto^i \mathbf{error}$. First, let us suppose that $\rho_1(\gamma_1(\mathbf{t}_1)) [\rho_1(\mathbf{F})] \mapsto^i \mathbf{t}'_1$. If \mathbf{t}'_1 is not a value the result holds immediately. Let us suppose $\mathbf{t}'_1 = \mathbf{v}'_1$. Hence, by inspection of the operational semantics, it follows that there exist $i_1 < i$, and \mathbf{v}_{11} such that $\rho_1(\gamma_1(\mathbf{t}_1)) [\rho_1(\mathbf{F})] \mapsto^{i_1} \mathbf{v}_{11} [\rho_1(\mathbf{F})]$. We instantiate the premise $\Delta; \Gamma \vdash \mathbf{t}_1 \preceq \mathbf{t}_2 : \forall \mathbf{X}. \mathbf{G}$ with n, ρ and γ to obtain that $(n, \rho_1(\gamma_1(\mathbf{t}_1)), \rho_2(\gamma_2(\mathbf{t}_2))) \in \mathcal{J}_\rho[\forall \mathbf{X}. \mathbf{G}]$. We instantiate $\mathcal{J}_\rho[\forall \mathbf{X}. \mathbf{G}]$ with i_1 and \mathbf{v}_{11} (note that $i_1 < i < n$), hence there exists \mathbf{v}_{12} , such that $\rho_2(\gamma_2(\mathbf{t}_2)) \xrightarrow{*} \mathbf{v}_{12}$, and $\triangleright^{i_1} (n, \mathbf{v}_{11}, \mathbf{v}_{12}) \in \mathcal{V}_\rho[\forall \mathbf{X}. \mathbf{G}]$. We instantiate $\mathcal{V}_\rho[\forall \mathbf{X}. \mathbf{G}]$ with $\mathbf{F}_1 = \rho_1(\mathbf{F})$, $\mathbf{F}_2 = \rho_2(\mathbf{F})$, and $R = \mathcal{V}_\rho[\mathbf{F}]$. Then $\triangleright^{i_1} (n, \mathbf{v}_{11} [\mathbf{F}_1], \mathbf{v}_{12} [\mathbf{F}_2]) \in \mathcal{J}_{\rho; X \mapsto (\mathbf{F}_1, \mathbf{F}_2, R)}[\mathbf{G}]$. We instantiate $\mathcal{J}_{\rho; X \mapsto (\mathbf{F}_1, \mathbf{F}_2, R)}[\mathbf{G}]$ with $i_2 = i - i_1$, and \mathbf{v}'_1 , such that $\mathbf{v}_{11} [\mathbf{F}_1] \mapsto^{i_2} \mathbf{v}'_1$. Therefore there must exist \mathbf{v}'_2 such that $\mathbf{v}_{12} [\mathbf{F}_2] \xrightarrow{*} \mathbf{v}'_2$ and $\triangleright^{i_1+i_2} (n, \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_{\rho; X \mapsto (\mathbf{F}_1, \mathbf{F}_2, R)}[\mathbf{G}]$. Finally, the result follows by Compositionality (Prop 6.9). \square

Lemma B.38 (Compositionality) *Let $\Delta \vdash \mathbf{F}$, $\Delta, \mathbf{X} \vdash \mathbf{G}$, $(n, \rho) \in \mathcal{D}[\Delta]$, and $R = \mathcal{V}_\rho[\mathbf{F}]$, then $\mathcal{V}_\rho[\mathbf{G}[\mathbf{F}/\mathbf{X}]] = \mathcal{V}_{\rho, X \mapsto (\rho_1(\mathbf{F}), \rho_2(\mathbf{F}), R)}[\mathbf{G}]$.*

PROOF. By case analysis on \mathbf{G} , and induction on (n, \mathbf{G}) . Let $\rho' = \rho, X \mapsto (\rho_1(\mathbf{F}), \rho_2(\mathbf{F}), R)$.

Case ($\mathbf{G} = \mathbf{B}$). We know that

$$\mathcal{V}_\rho[\mathbf{B}[\mathbf{F}/\mathbf{X}]] = \mathcal{V}_\rho[\mathbf{B}] = \mathcal{V}_{\rho'}[\mathbf{B}]$$

Therefore, the result holds.

Case ($\mathbf{G} = \mathbf{X}$). We know that

$$\mathcal{V}_\rho[\mathbf{X}[\mathbf{F}/\mathbf{X}]] = \mathcal{V}_\rho[\mathbf{F}] = R = \rho'.R(\mathbf{X}) = \mathcal{V}_{\rho'}[\mathbf{X}]$$

Therefore, the result holds.

Case ($\mathbf{G} = \mathbf{Y} \wedge \mathbf{Y} \not\rightsquigarrow \mathbf{X}$). We know that

$$\mathcal{V}_\rho[\mathbf{Y}[\mathbf{F}/\mathbf{X}]] = \mathcal{V}_\rho[\mathbf{Y}] = \rho.R(\mathbf{Y}) = \rho'.R(\mathbf{Y}) = \mathcal{V}_{\rho'}[\mathbf{Y}]$$

Therefore, the result holds.

Case ($\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$). We are required to prove that

$$\mathcal{V}_\rho[\mathbf{G}_1 \rightarrow \mathbf{G}_2[\mathbf{F}/\mathbf{X}]] = \mathcal{V}_{\rho, X \mapsto (\rho_1(\mathbf{F}), \rho_2(\mathbf{F}), R)}[\mathbf{G}_1 \rightarrow \mathbf{G}_2]$$

Or what is the same, we are required to prove that

$$\mathcal{V}_\rho[\mathbf{G}_1[\mathbf{F}/\mathbf{X}] \rightarrow \mathbf{G}_2[\mathbf{F}/\mathbf{X}]] = \mathcal{V}_{\rho, X \mapsto (\rho_1(\mathbf{F}), \rho_2(\mathbf{F}), R)}[\mathbf{G}_1 \rightarrow \mathbf{G}_2]$$

We divide this case in two proofs.

- If $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}_1[\mathbf{F}/\mathbf{X}] \rightarrow \mathbf{G}_2[\mathbf{F}/\mathbf{X}]]$ then $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_{\rho'}[\mathbf{G}_1 \rightarrow \mathbf{G}_2]$. We are required to prove that $\forall n' \leq n, \mathbf{v}'_1, \mathbf{v}'_2. \triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_{\rho'}[\mathbf{G}_1]$, we get that

$$(n', \mathbf{v}_1 \mathbf{v}'_1, \mathbf{v}_2 \mathbf{v}'_2) \in \mathcal{T}_{\rho'}[\mathbf{G}_2]$$

Since $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_{\rho'}[\mathbf{G}_1]$ by the induction hypothesis on (n', \mathbf{G}_1) , we know that $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[\mathbf{G}_1[\mathbf{F}/\mathbf{X}]]$. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}_1[\mathbf{F}/\mathbf{X}] \rightarrow \mathbf{G}_2[\mathbf{F}/\mathbf{X}]]$, we instantiate this with $n' \leq n, \mathbf{v}'_1, \mathbf{v}'_2. \triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[\mathbf{G}_1[\mathbf{F}/\mathbf{X}]]$. Therefore, we get that

$$(n', \mathbf{v}_1 \mathbf{v}'_1, \mathbf{v}_2 \mathbf{v}'_2) \in \mathcal{T}_\rho[\mathbf{G}_2[\mathbf{F}/\mathbf{X}]]$$

We know that for all $i < n'$, if $\mathbf{v}_1 \mathbf{v}'_1 \mapsto^i \mathbf{error}$, then $\mathbf{v}_2 \mathbf{v}'_2 \mapsto^* \mathbf{error}$, and therefore, the result holds. Also, we know that for all $i < n'$, if $\mathbf{v}_1 \mathbf{v}'_1 \mapsto^i \mathbf{v}_{1f}$, then we know that $\mathbf{v}_2 \mathbf{v}'_2 \mapsto^* \mathbf{v}_{2f}$ and $\triangleright^i (n', \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\mathbf{G}_2[\mathbf{F}/\mathbf{X}]]$. By the induction hypothesis on (n', \mathbf{G}_2) , we get that $\triangleright^i (n', \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho'}[\mathbf{G}_2]$, and the result holds.

- If $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_{\rho'}[\mathbf{G}_1 \rightarrow \mathbf{G}_2]$ then $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}_1[\mathbf{F}/\mathbf{X}] \rightarrow \mathbf{G}_2[\mathbf{F}/\mathbf{X}]]$.

We are required to prove that $\forall n' \leq n, \mathbf{v}'_1, \mathbf{v}'_2. \triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[\mathbf{G}_1[\mathbf{F}/\mathbf{X}]]$, we get that

$$(n', \mathbf{v}_1 \mathbf{v}'_1, \mathbf{v}_2 \mathbf{v}'_2) \in \mathcal{T}_\rho[\mathbf{G}_2[\mathbf{F}/\mathbf{X}]]$$

Since $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[\mathbf{G}_1[\mathbf{F}/\mathbf{X}]]$ by the induction hypothesis on \mathbf{G}_1 , we know that $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_{\rho'}[\mathbf{G}_1]$. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_{\rho'}[\mathbf{G}_1 \rightarrow \mathbf{G}_2]$, we instantiate this with $n' \leq n, \mathbf{v}'_1, \mathbf{v}'_2. \triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_{\rho'}[\mathbf{G}_1]$. Therefore, we get that

$$(n', \mathbf{v}_1 \mathbf{v}'_1, \mathbf{v}_2 \mathbf{v}'_2) \in \mathcal{T}_{\rho'}[\mathbf{G}_2]$$

We know that for all $i < n'$, if $\mathbf{v}_1 \mathbf{v}'_1 \mapsto^i \mathbf{error}$, then $\mathbf{v}_2 \mathbf{v}'_2 \mapsto^* \mathbf{error}$, and therefore, the result holds. Also, we know that for all $i < n'$, if $\mathbf{v}_1 \mathbf{v}'_1 \mapsto^i \mathbf{v}_{1f}$, then we know that $\mathbf{v}_2 \mathbf{v}'_2 \mapsto^* \mathbf{v}_{2f}$ and $\triangleright^i (n', \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho'}[\mathbf{G}_2]$. By the induction hypothesis on \mathbf{G}_2 , we get that $\triangleright^i (n', \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\mathbf{G}_2[\mathbf{F}/\mathbf{X}]]$, and the result holds.

Therefore, the result holds.

Case ($\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$). Similar to the function case.

Case ($\mathbf{G} = \forall \mathbf{Y}. \mathbf{G}_1$). We are required to prove that

$$\mathcal{V}_\rho[(\forall \mathbf{Y}. \mathbf{G}_1)[\mathbf{F}/\mathbf{X}]] = \mathcal{V}_{\rho, X \mapsto (\rho_1(\mathbf{F}), \rho_2(\mathbf{F}), R)}[\forall \mathbf{Y}. \mathbf{G}_1]$$

Or what is the same, we are required to prove that

$$\mathcal{V}_\rho[\forall \mathbf{Y}. (\mathbf{G}_1[\mathbf{F}/\mathbf{X}])] = \mathcal{V}_{\rho, X \mapsto (\rho_1(\mathbf{F}), \rho_2(\mathbf{F}), R)}[\forall \mathbf{Y}. \mathbf{G}_1]$$

We divide this case in two proofs.

- If $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\llbracket \forall \mathbf{Y}.(\mathbf{G}_1[\mathbf{F}/\mathbf{X}]) \rrbracket]$ then $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_{\rho'}[\llbracket \forall \mathbf{Y}.\mathbf{G}_1 \rrbracket]$. We are required to prove that for all $\vdash \mathbf{F}_1, \vdash \mathbf{F}_2, R \in \text{REL}[\mathbf{F}_1, \mathbf{F}_2]$, we get that

$$(n, \mathbf{v}_1[\mathbf{F}_1], \mathbf{v}_2[\mathbf{F}_2]) \in \mathcal{J}_{(\rho'; \mathbf{Y} \mapsto (\mathbf{F}_1, \mathbf{F}_2, R))}[\llbracket \mathbf{G}_1 \rrbracket]$$

Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\llbracket \forall \mathbf{Y}.(\mathbf{G}_1[\mathbf{F}/\mathbf{X}]) \rrbracket]$, we instantiate this with $\vdash \mathbf{F}_1, \vdash \mathbf{F}_2, R \in \text{REL}[\mathbf{F}_1, \mathbf{F}_2]$. Therefore, we get that

$$(n, \mathbf{v}_1[\mathbf{F}_1], \mathbf{v}_2[\mathbf{F}_2]) \in \mathcal{J}_{(\rho; \mathbf{Y} \mapsto (\mathbf{F}_1, \mathbf{F}_2, R))}[\llbracket \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \rrbracket]$$

We know that for all $i < n$, if $\mathbf{v}_1[\mathbf{F}_1] \mapsto^i \mathbf{error}$, then $\mathbf{v}_2[\mathbf{F}_2] \mapsto^* \mathbf{error}$, and therefore, the result holds. Also, we know that for all $i < n$, if $\mathbf{v}_1[\mathbf{F}_1] \mapsto^i \mathbf{v}_{1f}$, then we know that $\mathbf{v}_2[\mathbf{F}_2] \mapsto^* \mathbf{v}_{2f}$ and $\triangleright^i (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho; \mathbf{Y} \mapsto (\mathbf{F}_1, \mathbf{F}_2, R)}[\llbracket \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \rrbracket]$. By the induction hypothesis on \mathbf{G}_1 , we get that $\triangleright^i (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho'; \mathbf{Y} \mapsto (\mathbf{F}_1, \mathbf{F}_2, R)}[\llbracket \mathbf{G}_1 \rrbracket]$, and the result holds.

- If $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_{\rho'}[\llbracket \forall \mathbf{Y}.\mathbf{G}_1 \rrbracket]$ then $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\llbracket \forall \mathbf{Y}.(\mathbf{G}_1[\mathbf{F}/\mathbf{X}]) \rrbracket]$. We are required to prove that for all $\vdash \mathbf{F}_1, \vdash \mathbf{F}_2, R \in \text{REL}[\mathbf{F}_1, \mathbf{F}_2]$, we get that

$$(n, \mathbf{v}_1[\mathbf{F}_1], \mathbf{v}_2[\mathbf{F}_2]) \in \mathcal{J}_{(\rho; \mathbf{Y} \mapsto (\mathbf{F}_1, \mathbf{F}_2, R))}[\llbracket \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \rrbracket]$$

Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_{\rho'}[\llbracket \forall \mathbf{Y}.\mathbf{G}_1 \rrbracket]$, we instantiate this with $\vdash \mathbf{F}_1, \vdash \mathbf{F}_2, R \in \text{REL}[\mathbf{F}_1, \mathbf{F}_2]$. Therefore, we get that

$$(n, \mathbf{v}_1[\mathbf{F}_1], \mathbf{v}_2[\mathbf{F}_2]) \in \mathcal{J}_{(\rho'; \mathbf{Y} \mapsto (\mathbf{F}_1, \mathbf{F}_2, R))}[\llbracket \mathbf{G}_1 \rrbracket]$$

We know that for all $i < n$, if $\mathbf{v}_1[\mathbf{F}_1] \mapsto^i \mathbf{error}$, then $\mathbf{v}_2[\mathbf{F}_2] \mapsto^* \mathbf{error}$, and therefore, the result holds. Also, we know that for all $i < n$, if $\mathbf{v}_1[\mathbf{F}_1] \mapsto^i \mathbf{v}_{1f}$, then we know that $\mathbf{v}_2[\mathbf{F}_2] \mapsto^* \mathbf{v}_{2f}$ and $\triangleright^i (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho'; \mathbf{Y} \mapsto (\mathbf{F}_1, \mathbf{F}_2, R)}[\llbracket \mathbf{G}_1 \rrbracket]$. By the induction hypothesis on \mathbf{G}_1 , we get that $\triangleright^i (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho; \mathbf{Y} \mapsto (\mathbf{F}_1, \mathbf{F}_2, R)}[\llbracket \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \rrbracket]$, and the result holds.

Therefore, the result holds.

Case ($\mathbf{G} = ?_\delta$). Then we have to prove that

$$\mathcal{V}_\rho[\llbracket ?_{[\mathbf{F}/\mathbf{X}]\delta} \rrbracket] = \mathcal{V}_{\rho, X \mapsto (\rho_1(\mathbf{F}), \rho_2(\mathbf{F}), R)}[\llbracket ?_\delta \rrbracket]$$

If $\mathbf{X} \notin \text{FTV}(\delta)$ ($\Delta \vdash ?_\delta$) then $?_{[\mathbf{F}/\mathbf{X}]\delta} = ?_\delta$. Therefore, we are required to prove that $\mathcal{V}_\rho[\llbracket ?_\delta \rrbracket] = \mathcal{V}_{\rho'}[\llbracket ?_\delta \rrbracket]$, which follows immediately, by Lemma B.46. If $\mathbf{X} \in \text{FTV}(\delta)$, we know that $\mathbf{X} : \mathbf{X} \in \delta$. Let us prove that $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\llbracket ?_{[\mathbf{F}/\mathbf{X}]\delta} \rrbracket]$ if and only if $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_{\rho'}[\llbracket ?_\delta \rrbracket]$. Let $\mathbf{F}_1 = \rho_1(\mathbf{F})$ and $\mathbf{F}_2 = \rho_2(\mathbf{F})$.

- First let us prove that $(n, \mathbf{v}_1, \mathbf{v}_2) \in \text{ATOM}_\rho^v[\llbracket ?_{[\mathbf{F}/\mathbf{X}]\delta} \rrbracket]$ if and only if $(n, \mathbf{v}_1, \mathbf{v}_2) \in \text{ATOM}_{\rho'}^v[\llbracket ?_\delta \rrbracket]$. We know that

$$\begin{aligned} \text{ATOM}_\rho^v[\llbracket ?_{[\mathbf{F}/\mathbf{X}]\delta} \rrbracket] &= \text{ATOM}[\rho_1(?_{[\mathbf{F}/\mathbf{X}]\delta}), \rho_2(?_{[\mathbf{F}/\mathbf{X}]\delta})] \\ &= \text{ATOM}[\rho_1(?_{[\rho_1(\mathbf{F})/\mathbf{X}]\delta}), \rho_2(?_{[\rho_2(\mathbf{F})/\mathbf{X}]\delta})] \end{aligned}$$

and

$$\begin{aligned} \text{ATOM}_{\rho'}^v[\llbracket ?_\delta \rrbracket] &= \text{ATOM}[(\rho_1, X \mapsto \rho_1(\mathbf{F}))(?_\delta), (\rho_2, X \mapsto \rho_2(\mathbf{F}))(?_\delta)] \\ &= \text{ATOM}[\rho_1(?_\delta[\rho_1(\mathbf{F})/\mathbf{X}]), \rho_2(?_\delta[\rho_2(\mathbf{F})/\mathbf{X}])] \\ &= \text{ATOM}[\rho_1(?_{[\rho_1(\mathbf{F})/\mathbf{X}]\delta}), \rho_2(?_{[\rho_2(\mathbf{F})/\mathbf{X}]\delta})] \end{aligned}$$

and the result holds.

- Let $\delta' = [\mathbf{F}/\mathbf{X}]\delta$. Note that since $\delta' = [\mathbf{F}/\mathbf{X}]\delta$ and $\delta' \vdash \mathbf{F}$, we know that

$$\rho_i(\delta') = \rho_i([\mathbf{F}/\mathbf{X}]\delta) = (\rho, X \mapsto (\rho_1(\mathbf{F}), \rho_2(\mathbf{F}), R))_i(\delta) = \rho'_i(\delta)$$

Next we prove that

$\forall \mathbf{G}_R$ and $\vdash \varepsilon : \delta' \rightarrow \mathbf{G}_R$, $(n, \rho_1(\varepsilon)\mathbf{v}_1 :: \rho_1(\mathbf{G}_R), \rho_2(\varepsilon)\mathbf{v}_2 :: \rho_2(\mathbf{G}_R)) \in \mathcal{T}_\rho[\mathbf{G}_R]$, if and only if $\forall \mathbf{G}'_R$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}'_R$, $(n, \rho'_1(\varepsilon')\mathbf{v}_1 :: \rho'_1(\mathbf{G}'_R), \rho'_2(\varepsilon')\mathbf{v}_2 :: \rho'_2(\mathbf{G}'_R)) \in \mathcal{T}_{\rho'}[\mathbf{G}'_R]$.

Let us prove the \Leftarrow direction first. We proceed by case analysis on \mathbf{G}_R . We proceed as follows, for each \mathbf{G}_R and $\vdash \varepsilon : \delta \rightarrow \mathbf{G}_R$, we show that exists some \mathbf{G}'_R and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}'_R$ such that $\rho_i(\varepsilon) = \rho'_i(\varepsilon')$.

1. (Case $\mathbf{G}_R = \mathbf{B}$). Therefore, $\varepsilon = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})]$. Then $\rho_i(\varepsilon) = \rho_i([(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})]) = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})]$. Next, we can choose $\mathbf{G}'_R = \mathbf{B}$, and $\varepsilon' = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})]$. Note that $\rho'_i(\varepsilon') = \rho'_i([(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})]) = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})] = \rho_i(\varepsilon)$. Thus, the result holds.
2. (Case $\mathbf{G}_R = \mathbf{Y}$ and $\mathbf{Y} \not\prec \mathbf{X}$). Since $\delta' = [\mathbf{F}/\mathbf{X}]\delta$ and $\mathbf{G}_R = \mathbf{Y} \dashv \delta'$, we know that $\mathbf{Y} \not\prec \mathbf{X}$, $\mathbf{Y} : (\mathbf{F}'[\mathbf{F}/\mathbf{X}]) \in \delta'$, $\mathbf{Y} : \mathbf{F}' \in \delta$. Therefore, $\varepsilon = [((\mathbf{F}'[\mathbf{F}/\mathbf{X}]), \text{inj}_{\mathbf{Y}}, (\mathbf{F}'[\mathbf{F}/\mathbf{X}]))]$. Then

$$\begin{aligned} \rho_i(\varepsilon) &= \rho_i([((\mathbf{F}'[\mathbf{F}/\mathbf{X}]), \text{inj}_{\mathbf{Y}}, (\mathbf{F}'[\mathbf{F}/\mathbf{X}])))]) = [(\rho_i(\mathbf{F}'[\mathbf{F}/\mathbf{X}]), \text{inj}_{\mathbf{Y}}, \rho_i(\mathbf{F}'[\mathbf{F}/\mathbf{X}]))] = \\ &= [(\rho'_i(\mathbf{F}'), \text{inj}_{\mathbf{Y}}, \rho_i(\mathbf{F}'))] \end{aligned}$$

Next, we can choose $\mathbf{G}'_R = \mathbf{Y} \dashv \delta$, and $\varepsilon' = [(\mathbf{F}', \text{inj}_{\mathbf{Y}}, \mathbf{F}')]$. Then

$$\rho'_i(\varepsilon') = \rho'_i([(\mathbf{F}', \text{inj}_{\mathbf{Y}}, \mathbf{F}')]]) = [(\rho'_i(\mathbf{F}'), \text{inj}_{\mathbf{Y}}, \rho'_i(\mathbf{F}'))] = \rho_i(\varepsilon)$$

Thus, the result holds.

3. (Case $\mathbf{G}_R = \mathbf{X}$). Since $\mathbf{X} : \mathbf{F} \in \delta'$, we know that $\varepsilon = [(\mathbf{F}, \text{inj}_{\mathbf{X}}, \mathbf{F})]$. Note that

$$\rho_i(\varepsilon) = \rho_i([(\mathbf{F}, \text{inj}_{\mathbf{X}}, \mathbf{F})]) = [(\rho_i(\mathbf{F}), \text{inj}_{\mathbf{X}}, \rho_i(\mathbf{F}))]$$

Next, we can choose $\mathbf{G}'_R = \mathbf{X}$, and since $\mathbf{X} : \mathbf{X} \in \delta$, we get that $\varepsilon' = [(\mathbf{X}, \text{inj}_{\mathbf{X}}, \mathbf{X})]$. Then

$$\rho'_i(\varepsilon') = \rho'_i([(\mathbf{X}, \text{inj}_{\mathbf{X}}, \mathbf{X})]) = [(\rho'_i(\mathbf{X}), \rho'_i(\text{inj}_{\mathbf{X}}), \rho'_i(\mathbf{X}))] = [(\rho_i(\mathbf{F}), \text{inj}_{\mathbf{X}}, \rho_i(\mathbf{F}))] = \rho_i(\varepsilon)$$

Thus, the result holds.

4. (Case $\mathbf{G}_R = ?_{\delta'} \rightarrow ?_{\delta'}$). Then $\varepsilon = [(?_{\delta'} \rightarrow ?_{\delta'}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})]$. Then $\rho_i(\varepsilon) = [(?_{\rho_i(\delta')} \rightarrow ?_{\rho_i(\delta')}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})]$. Next, we can choose $\mathbf{G}'_R = ?_{\delta} \rightarrow ?_{\delta}$, and $\varepsilon' = [(?_{\delta} \rightarrow ?_{\delta}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})]$. Then, since $\rho_i(\delta') = \rho'_i(\delta)$, we know that

$$\begin{aligned} \rho'_i(\varepsilon') &= [(?_{\rho'_i(\delta)} \rightarrow ?_{\rho'_i(\delta)}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})] \\ &= [(?_{\rho_i(\delta')} \rightarrow ?_{\rho_i(\delta')}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})] \\ &= \rho_i(\varepsilon) \end{aligned}$$

and the result holds.

5. (Case $\mathbf{G}_R = ?_{\delta'} \times ?_{\delta'}$). Then $\varepsilon = [(?_{\delta'} \times ?_{\delta'}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})]$. Then $\rho_i(\varepsilon) = [(?_{\rho_i(\delta')} \times ?_{\rho_i(\delta')}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})]$. Next, we can choose $\mathbf{G}'_R =$

$?_{\delta} \times ?_{\delta}$, and $\varepsilon' = [(?_{\delta} \times ?_{\delta}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})]$. Then, since $\rho_i(\delta') = \rho'_i(\delta)$, we know that

$$\begin{aligned}\rho'_i(\varepsilon') &= [(?_{\rho'_i(\delta)} \times ?_{\rho'_i(\delta)}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})] \\ &= [(?_{\rho_i(\delta')} \times ?_{\rho_i(\delta')}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})] \\ &= \rho_i(\varepsilon)\end{aligned}$$

and the result holds.

6. (Case $\mathbf{G}_R = \forall \mathbf{Y}. ?_{\delta', \mathbf{Y}: \mathbf{Y}}$). Note that $\varepsilon = [(\forall \mathbf{Y}. ?_{\delta', \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})]$. Then

$$\rho_i(\varepsilon) = \rho_i([(\forall \mathbf{Y}. ?_{\delta', \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})]) = [(\forall \mathbf{Y}. ?_{\rho'_i(\delta'), \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})]$$

Next, we can choose $\mathbf{G}'_R = \forall \mathbf{Y}. ?_{\delta, \mathbf{Y}: \mathbf{Y}}$, and $\varepsilon' = [(\forall \mathbf{Y}. ?_{\delta, \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})]$. Then, since $\rho_i(\delta') = \rho'_i(\delta)$, we know that

$$\rho_i(\varepsilon') = \rho'_i([(\forall \mathbf{Y}. ?_{\delta, \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})]) = [(\forall \mathbf{Y}. ?_{\rho'_i(\delta), \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})] =$$

and the result holds.

Let us prove the \Rightarrow direction. We proceed by case analysis on \mathbf{G}'_R . We proceed as follows, for each $\mathbf{G}'_R \vdash \delta$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}'_R$, we show that exists some $\mathbf{G}_R \vdash \delta'$ and $\vdash \varepsilon : \delta' \rightarrow \mathbf{G}_R$ such that $\rho'_i(\varepsilon') = \rho_i(\varepsilon)$.

- (Case $\mathbf{G}'_R = \mathbf{B}$). Therefore, $\varepsilon' = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})]$. Then $\rho'_i(\varepsilon') = \rho'_i([(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})]) = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})]$. Next, we can choose $\mathbf{G}_R = \mathbf{B} \vdash \delta'$, and $\varepsilon = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})]$. Note that $\rho_i(\varepsilon) = \rho_i([(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})]) = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})] = \rho'_i(\varepsilon')$. Thus, the result holds.
- (Case $\mathbf{G}'_R = \mathbf{Y}$ and $\mathbf{Y} \not\leftrightarrow \mathbf{X}$). Since $\delta' = [\mathbf{F}/\mathbf{X}]\delta$ and $\mathbf{G}'_R = \mathbf{Y} \vdash \delta$, we know that $\mathbf{Y} : \mathbf{F}' \in \delta$ and $\mathbf{Y} : (\mathbf{F}'[\mathbf{F}/\mathbf{X}]) \in \delta'$. Therefore, $\varepsilon' = [(\mathbf{F}', \text{inj}_{\mathbf{Y}}, \mathbf{F}')] =$

$$\rho'_i(\varepsilon') = \rho'_i([(\mathbf{F}', \text{inj}_{\mathbf{Y}}, \mathbf{F}')] = [(\rho'_i(\mathbf{F}'), \text{inj}_{\mathbf{Y}}, \rho'_i(\mathbf{F}'))] = [(\rho_i(\mathbf{F}'[\mathbf{F}/\mathbf{X}]), \text{inj}_{\mathbf{Y}}, \rho_i(\mathbf{F}'[\mathbf{F}/\mathbf{X}]))]$$

Next, we can choose $\mathbf{G}_R = \mathbf{Y} \vdash \delta'$, and $\varepsilon = [((\mathbf{F}'[\mathbf{F}/\mathbf{X}]), \text{inj}_{\mathbf{Y}}, (\mathbf{F}'[\mathbf{F}/\mathbf{X}]))]$. Note that $\rho_i(\varepsilon) = \rho_i([((\mathbf{F}'[\mathbf{F}/\mathbf{X}]), \text{inj}_{\mathbf{Y}}, (\mathbf{F}'[\mathbf{F}/\mathbf{X}]))]) = [(\rho_i(\mathbf{F}'[\mathbf{F}/\mathbf{X}]), \text{inj}_{\mathbf{Y}}, \rho_i(\mathbf{F}'[\mathbf{F}/\mathbf{X}]))] = \rho'_i(\varepsilon')$

Thus, the result holds.

- (Case $\mathbf{G}'_R = \mathbf{X}$). Since $\mathbf{X} : \mathbf{X} \in \delta$, we know that $\varepsilon' = [(\mathbf{X}, \text{inj}_{\mathbf{X}}, \mathbf{X})]$. Then

$$\rho'_i(\varepsilon') = \rho'_i([(\mathbf{X}, \text{inj}_{\mathbf{X}}, \mathbf{X})]) = [(\rho'_i(\mathbf{X}), \rho'_i(\text{inj}_{\mathbf{X}}), \rho'_i(\mathbf{X}))] = [(\rho_i(\mathbf{F}), \text{inj}_{\mathbf{X}}, \rho_i(\mathbf{F}))]$$

Next, we can choose $\mathbf{G}_R = \mathbf{X}$ and since $\mathbf{X} : \mathbf{F} \in \delta'$, we know that $\varepsilon = [(\mathbf{F}, \text{inj}_{\mathbf{X}}, \mathbf{F})]$. Note that

$$\rho_i(\varepsilon) = \rho_i([(\mathbf{F}, \text{inj}_{\mathbf{X}}, \mathbf{F})]) = [(\rho_i(\mathbf{F}), \text{inj}_{\mathbf{X}}, \rho_i(\mathbf{F}))] = \rho'_i(\varepsilon')$$

Thus, the result holds.

- (Case $\mathbf{G}'_R = ?_{\delta} \rightarrow ?_{\delta}$). Then $\varepsilon' = [(?_{\delta} \rightarrow ?_{\delta}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})]$. Thus, $\rho'_i(\varepsilon') = [(?_{\rho'_i(\delta)} \rightarrow ?_{\rho'_i(\delta)}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})]$. Next, we can choose $\mathbf{G}_R = ?_{\delta'} \rightarrow ?_{\delta'}$, and $\varepsilon = [(?_{\delta'} \rightarrow ?_{\delta'}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})]$.

Thus, $\rho_i(\varepsilon) = [(?_{\rho_i(\delta')} \rightarrow ?_{\rho_i(\delta')}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})]$. Then, since $\rho_i(\delta') = \rho'_i(\delta)$, we know that

$$\begin{aligned}\rho'_i(\varepsilon') &= [(?_{\rho'_i(\delta)} \rightarrow ?_{\rho'_i(\delta)}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})] \\ &= [(?_{\rho_i(\delta')} \rightarrow ?_{\rho_i(\delta')}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})] \\ &= \rho_i(\varepsilon)\end{aligned}$$

and the result holds.

5. (Case $\mathbf{G}'_{\mathbf{R}} = ?_{\delta} \times ?_{\delta}$). Then $\varepsilon' = [(?_{\delta} \times ?_{\delta}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})]$. Thus, $\rho'_i(\varepsilon') = [(?_{\rho'_i(\delta)} \times ?_{\rho'_i(\delta)}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})]$. Next, we can choose $\mathbf{G}_{\mathbf{R}} = ?_{\delta'} \times ?_{\delta'}$, and $\varepsilon = [(?_{\delta'} \times ?_{\delta'}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})]$. Thus, $\rho_i(\varepsilon) = [(?_{\rho_i(\delta')} \times ?_{\rho_i(\delta')}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})]$. Then, since $\rho_i(\delta') = \rho'_i(\delta)$, we know that

$$\begin{aligned} \rho'_i(\varepsilon') &= [(?_{\rho'_i(\delta)} \times ?_{\rho'_i(\delta)}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})] \\ &= [(?_{\rho_i(\delta')} \times ?_{\rho_i(\delta')}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})] \\ &= \rho_i(\varepsilon) \end{aligned}$$

and the result holds.

6. (Case $\mathbf{G}'_{\mathbf{R}} = \forall \mathbf{Y}. ?_{\delta, \mathbf{Y}: \mathbf{Y}}$). Note that $\varepsilon' = [(\forall \mathbf{Y}. ?_{\delta, \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})]$. Thus, $\rho_i(\varepsilon') = \rho'_i([(\forall \mathbf{Y}. ?_{\delta, \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})]) = [(\forall \mathbf{Y}. ?_{\rho'_i(\delta), \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})]$. Next, we can choose $\mathbf{G}_{\mathbf{R}} = \forall \mathbf{Y}. ?_{\delta', \mathbf{Y}: \mathbf{Y}}$, and $\varepsilon = [(\forall \mathbf{Y}. ?_{\delta', \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})]$. Thus,

$$\rho_i(\varepsilon) = \rho_i([(\forall \mathbf{Y}. ?_{\delta', \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})]) = [(\forall \mathbf{Y}. ?_{\rho_i(\delta'), \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})]$$

Then, since $\rho_i(\delta') = \rho'_i(\delta)$, we know that

$$\rho_i(\varepsilon') = \rho'_i([(\forall \mathbf{Y}. ?_{\delta, \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})]) = [(\forall \mathbf{Y}. ?_{\rho'_i(\delta), \mathbf{Y}: \mathbf{Y}}, \text{inj}_{\forall}(\forall \mathbf{Y}. \text{inj}_{?}), \forall \mathbf{Y}. \text{inj}_{?})] =$$

and the result holds. □

Lemma B.39 (Ascription Lemma) *If $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_{\rho}[\mathbf{G}]$, $(n, \rho) \in \mathcal{D}[\Delta]$, $\Delta \vdash \mathbf{G}'$ and $\varepsilon : \mathbf{G} \sim \mathbf{G}'$, then $(n, \rho_1(\varepsilon)\mathbf{v}_1 :: \rho_1(\mathbf{G}'), \rho_2(\varepsilon)\mathbf{v}_2 :: \rho_2(\mathbf{G}')) \in \mathcal{T}_{\rho}[\mathbf{G}']$.*

PROOF. We proceed by induction on $(n, \mathbf{G}, \mathbf{G}')$. We start by doing a case analysis on $\varepsilon : \mathbf{G} \sim \mathbf{G}'$. Suppose $\mathbf{v}_i = \varepsilon_{\mathbf{ui}} \mathbf{u}_i :: \rho_i(\mathbf{G}')$.

Case $(\varepsilon : \mathbf{B} \sim \mathbf{B})$. We know that

1. Since $\mathbf{G} = \mathbf{B}$ and $\mathbf{G}' = \mathbf{B}$, by Lemma B.41, we know that $\varepsilon = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\} = \rho_i(\varepsilon)$.
2. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_{\rho}[\mathbf{B}]$, we know that $(n, \mathbf{v}_1, \mathbf{v}_2) \in \text{ATOM}_{\rho}[\mathbf{B}]$, and therefore, $\mathbf{v}_i = \varepsilon_{\mathbf{ui}} \mathbf{u}_i :: \mathbf{B}$, where $\varepsilon_{\mathbf{ui}} = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\}$, by Lemma B.41.
3. Then $\varepsilon_{\mathbf{ui}} \circ \rho_i(\varepsilon) = \varepsilon_{\mathbf{ui}} = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\}$.
4. We have to prove that $(n, \varepsilon_{\mathbf{u1}} \mathbf{u}_1 :: \mathbf{B}, \varepsilon_{\mathbf{u2}} \mathbf{u}_2 :: \mathbf{B}) \in \mathcal{V}_{\rho}[\mathbf{B}]$, which follows by Lemma B.45.

Case $(\varepsilon : \mathbf{X} \sim \mathbf{X})$. We know that

1. Since $\mathbf{G} = \mathbf{X}$ and $\mathbf{G}' = \mathbf{X}$, by Lemma B.41, we know that $\varepsilon = \{(\mathbf{X}, \mathbf{X}, \mathbf{X})\}$, and therefore, $\rho_i(\varepsilon) = \rho_i(\{(\mathbf{X}, \mathbf{X}, \mathbf{X})\}) = [(\rho_i(\mathbf{X}), \text{refl}_{\rho_i(\mathbf{X})}^{\sqsubseteq}, \text{refl}_{\rho_i(\mathbf{X})}^{\sqsubseteq})] = \text{reflEv}(\rho_i(\mathbf{X}))$.
2. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_{\rho}[\mathbf{X}]$, we know that $(n, \mathbf{v}_1, \mathbf{v}_2) \in \text{ATOM}[\rho_1(\mathbf{X}), \rho_2(\mathbf{X})]$, and therefore, $\mathbf{v}_i = \varepsilon_{\mathbf{ui}} \mathbf{u}_i :: \rho_i(\mathbf{X})$ and $\varepsilon_{\mathbf{ui}} : \mathbf{G}_{\mathbf{ui}} \sim \rho_i(\mathbf{X})$, for some $\mathbf{G}_{\mathbf{ui}}$.

3. By Lemma B.62, we know that $\varepsilon_{\mathbf{u}i} \circ \rho_i(\varepsilon) = \varepsilon_{\mathbf{u}i}$.
4. We have to prove that $(n, \varepsilon_{\mathbf{u}1} \mathbf{u}_1 \circ \rho_1(\mathbf{X}), \varepsilon_{\mathbf{u}2} \mathbf{u}_2 \circ \rho_2(\mathbf{X})) \in \mathcal{V}_\rho[\mathbf{X}]$, which follows by Lemma B.45.

Case $(\varepsilon : \mathbf{G}'_1 \rightarrow \mathbf{G}'_2 \sim \mathbf{G}''_1 \rightarrow \mathbf{G}''_2)$. We know that

1. $\mathbf{G} = \mathbf{G}''_1 \rightarrow \mathbf{G}''_2$, and $\mathbf{G}' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$ for some $\mathbf{G}''_i, \mathbf{G}'_i$.
2. We know that $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}''_1 \rightarrow \mathbf{G}''_2]$, where $\mathbf{v}_i = \varepsilon_i(\lambda \mathbf{x} : \mathbf{G}_{\mathbf{x}i} \cdot \mathbf{t}_i) \circ \mathbf{G}''_{i1} \rightarrow \mathbf{G}''_{i2}$, for $\mathbf{G}''_{i1} = \rho_i(\mathbf{G}''_1)$, $\mathbf{G}''_{i2} = \rho_i(\mathbf{G}''_2)$ and $\varepsilon_i : \mathbf{G}_i \sim \mathbf{G}''_{i1} \rightarrow \mathbf{G}''_{i2}$.
3. We have to prove that:

$$(n, \rho_1(\varepsilon) \mathbf{v}_1 \circ \mathbf{G}'_{11} \rightarrow \mathbf{G}'_{12}, \rho_2(\varepsilon) \mathbf{v}_2 \circ \mathbf{G}'_{21} \rightarrow \mathbf{G}'_{22}) \in \mathcal{J}_\rho[\mathbf{G}'_1 \rightarrow \mathbf{G}'_2]$$

for $\mathbf{G}'_{i1} = \rho_i(\mathbf{G}'_1)$ and $\mathbf{G}'_{i2} = \rho_i(\mathbf{G}'_2)$.

4. By Lemma B.40 and B.51, we know that $\varepsilon_i = \text{dom}(\varepsilon_i) \rightarrow \text{cod}(\varepsilon_i)$ and $\rho_i(\varepsilon) = \text{dom}(\rho_i(\varepsilon)) \rightarrow \text{cod}(\rho_i(\varepsilon))$.
5. As $\varepsilon_i \circ \rho_i(\varepsilon) = (\text{dom}(\rho_i(\varepsilon)) \circ \text{dom}(\varepsilon_i)) \rightarrow (\text{cod}(\varepsilon_i) \circ \text{cod}(\rho_i(\varepsilon)))$, this means that proving that

$$(\varepsilon_i \circ \rho_i(\varepsilon)) \text{ is defined} \iff (\varepsilon_2 \circ \rho_2(\varepsilon)) \text{ is defined}$$

is equivalent to prove that

$$\begin{aligned} (\text{dom}(\rho_1(\varepsilon)) \circ \text{dom}(\varepsilon_1)) \text{ is defined} &\iff (\text{dom}(\rho_2(\varepsilon)) \circ \text{dom}(\varepsilon_2)) \text{ is defined} \wedge \\ (\text{cod}(\varepsilon_1) \circ \text{cod}(\rho_1(\varepsilon))) \text{ is defined} &\iff (\text{cod}(\varepsilon_2) \circ \text{cod}(\rho_2(\varepsilon))) \text{ is defined} \end{aligned}$$

6. Suppose that $\varepsilon_i \circ \rho_i(\varepsilon) = \varepsilon_{\lambda i}$. Then we have to prove that

$$(n, \varepsilon_{\lambda 1}(\lambda \mathbf{x} : \mathbf{G}_{\mathbf{x}1} \cdot \mathbf{t}_1) \circ \mathbf{G}'_{11} \rightarrow \mathbf{G}'_{12}, \varepsilon_{\lambda 2}(\lambda \mathbf{x} : \mathbf{G}_{\mathbf{x}2} \cdot \mathbf{t}_2) \circ \mathbf{G}'_{21} \rightarrow \mathbf{G}'_{22}) \in \mathcal{V}_\rho[\mathbf{G}'_1 \rightarrow \mathbf{G}'_2]$$

i.e. let us take any $n' \leq n$, \mathbf{v}'_1 and \mathbf{v}'_2 , such that $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[\mathbf{G}'_1]$, where $\mathbf{v}'_i = \varepsilon'_{i1} \mathbf{u}'_i \circ \mathbf{G}'_{i1}$, then we have to prove that

$$(n', (\varepsilon_{\lambda 1}(\lambda \mathbf{x} : \mathbf{G}_{\mathbf{x}1} \cdot \mathbf{t}_1) \circ \mathbf{G}'_{11} \rightarrow \mathbf{G}'_{12}) \mathbf{v}'_1, (\varepsilon_{\lambda 2}(\lambda \mathbf{x} : \mathbf{G}_{\mathbf{x}2} \cdot \mathbf{t}_2) \circ \mathbf{G}'_{21} \rightarrow \mathbf{G}'_{22}) \mathbf{v}'_2) \in \mathcal{J}_\rho[\mathbf{G}'_2]$$

7. By Lemma B.47, we know that

- $\text{dom}(\varepsilon_{\lambda i}) = \text{dom}(\varepsilon_i \circ \rho_i(\varepsilon)) = \text{dom}(\rho_i(\varepsilon)) \circ \text{dom}(\varepsilon_i)$
- $\text{cod}(\varepsilon_{\lambda i}) = \text{cod}(\varepsilon_i \circ \rho_i(\varepsilon)) = \text{cod}(\varepsilon_i) \circ \text{cod}(\rho_i(\varepsilon))$

8. By induction hypothesis on $(n' - 1, \mathbf{G}'_1, \mathbf{G}''_1)$, with $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[\mathbf{G}'_1]$ and $\text{dom}(\varepsilon) : \mathbf{G}'_1 \sim \mathbf{G}''_1$, we obtain that

$$\triangleright (n', \text{dom}(\rho_1(\varepsilon)) \mathbf{v}'_1 \circ \mathbf{G}''_{11}, \text{dom}(\rho_2(\varepsilon)) \mathbf{v}'_2 \circ \mathbf{G}''_{21}) \in \mathcal{J}_\rho[\mathbf{G}''_1]$$

9. If $(\varepsilon'_i \circ \text{dom}(\rho_i(\varepsilon)))$ fails, then $(\varepsilon'_2 \circ \text{dom}(\rho_2(\varepsilon)))$ fails, and the result holds.
10. If $(\varepsilon'_i \circ \text{dom}(\rho_i(\varepsilon)))$ does not fail, i.e. $(\varepsilon'_1 \circ \text{dom}(\rho_1(\varepsilon))) = \varepsilon_{\mathbf{a}1}$ and $(\varepsilon'_2 \circ \text{dom}(\rho_2(\varepsilon))) = \varepsilon_{\mathbf{a}2}$, then it is true that:

$$\triangleright (n', \varepsilon_{\mathbf{a}1} \mathbf{u}'_1 \circ \mathbf{G}''_{11}, \varepsilon_{\mathbf{a}2} \mathbf{u}'_2 \circ \mathbf{G}''_{21}) \in \mathcal{V}_\rho[\mathbf{G}''_1]$$

11. By the definition of $\mathcal{V}_\rho[\mathbf{G}_1'' \longrightarrow \mathbf{G}_2'']$, we know that:

$$\forall n'' \leq n. \forall \mathbf{v}_1'', \mathbf{v}_2''. \triangleright (n'', \mathbf{v}_1'', \mathbf{v}_2'') \in \mathcal{V}_\rho[\mathbf{G}_1''] \Rightarrow (n'', \mathbf{v}_1 \mathbf{v}_1'', \mathbf{v}_2 \mathbf{v}_2'') \in \mathcal{T}_\rho[\mathbf{G}_2'']$$

We instantiate this with $v_i'' = \varepsilon_{ai} u_i' :: \mathbf{G}_{i1}''$ and $n'' = n' \leq n$, to derive that:

$$(n', (\varepsilon_1(\lambda \mathbf{x} : \mathbf{G}_{x1}. \mathbf{t}_1) :: \mathbf{G}_{11}'' \longrightarrow \mathbf{G}_{12}'') (\varepsilon_{a1} u_1' :: \mathbf{G}_{11}''), (\varepsilon_2(\lambda \mathbf{x} : \mathbf{G}_{x2}. \mathbf{t}_2) :: \mathbf{G}_{21}'' \longrightarrow \mathbf{G}_{22}'') (\varepsilon_{a2} u_2' :: \mathbf{G}_{21}'')) \in \mathcal{V}_\rho[\mathbf{G}_2'']$$

12. Then by Lemma B.18, as $(\varepsilon_1' ; \text{dom}(\rho_i(\varepsilon))) ; \text{dom}(\varepsilon_i) = \varepsilon_1''' \iff \varepsilon_1' ; (\text{dom}(\rho_i(\varepsilon)) ; \text{dom}(\varepsilon_i)) = \varepsilon_1'''$, then if $(\text{dom}(\rho_1(\varepsilon)) ; \text{dom}(\varepsilon_1))$ is not defined and $(\text{dom}(\rho_2(\varepsilon)) ; \text{dom}(\varepsilon_2))$ is defined, we get a contradiction as both must behave uniformly as the terms belong to $\mathcal{T}_\rho[\mathbf{G}_2'']$. Then if both combination of evidence fail, then the result follows immediately. Let us suppose that the combination does not fail, then

$$(\varepsilon_i(\lambda \mathbf{x} : \mathbf{G}_{xi}. \mathbf{t}_i) :: \mathbf{G}_{i1}'' \longrightarrow \mathbf{G}_{i2}'') (\varepsilon_{ai} u_i' :: \mathbf{G}_{i1}'') \xrightarrow{1} \text{cod}(\varepsilon_i) ([\varepsilon_i''' u_i' :: \mathbf{G}_{xi}]/\mathbf{x}] \mathbf{t}_i) :: \mathbf{G}_{i2}''$$

where $(\varepsilon_{a1} ; \text{dom}(\varepsilon_1)) = \varepsilon_1'''$ and $(\varepsilon_{a2} ; \text{dom}(\varepsilon_2)) = \varepsilon_2'''$. If the resulting terms reduce to values (\mathbf{v}_{if})

$$\begin{aligned} \text{cod}(\varepsilon_1) ([\varepsilon_1''' u_1' :: \mathbf{G}_{x1}]/\mathbf{x}] \mathbf{t}_1) :: \mathbf{G}_{12}'' &\longrightarrow^k \mathbf{v}_{1f} \\ \text{cod}(\varepsilon_1) ([\varepsilon_2''' u_2' :: \mathbf{G}_{x2}]/\mathbf{x}] \mathbf{t}_2) :: \mathbf{G}_{22}'' &\longrightarrow^* \mathbf{v}_{2f} \end{aligned}$$

then $\triangleright^{1+k} (n', \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\mathbf{G}_2'']$.

13. We apply induction hypothesis to the previous result with $(n' - 1 - k, \mathbf{G}_2', \mathbf{G}_2')$, with evidence $\text{cod}(\varepsilon) : \mathbf{G}_2'' \sim \mathbf{G}_2'$, to obtain that:

$$\triangleright^{1+k} (n', \text{cod}(\rho_i(\varepsilon)) \mathbf{v}_{1f} :: \mathbf{G}_2', \text{cod}(\rho_i(\varepsilon)) \mathbf{v}_{2f} :: \mathbf{G}_2') \in \mathcal{T}_\rho[\mathbf{G}_2']$$

Then \mathbf{v}_{if} has to have the form: $\mathbf{v}_{if} = \varepsilon_{ci} u_{if}' :: \mathbf{G}_{i2}'$, where $\varepsilon_1''' ; \text{cod}(\varepsilon_1) = \varepsilon_{c1}'$, and $\varepsilon_2''' ; \text{cod}(\varepsilon_2) = \varepsilon_{c2}'$, for some ε_i'' , u_{if}' . Suppose that $\text{cod}(\varepsilon_1) ; \text{cod}(\rho_1(\varepsilon)) = \varepsilon_{z1}'$ is defined for now. Then by Lemma B.18, we know that $(\varepsilon_1'' ; \text{cod}(\varepsilon_1)) ; \text{cod}(\rho_1(\varepsilon)) = \varepsilon_{f1}' \iff \varepsilon_1'' ; (\text{cod}(\varepsilon_1) ; \text{cod}(\rho_1(\varepsilon))) = \varepsilon_{f1}'$, then $(\text{cod}(\varepsilon_i) ; \text{cod}(\rho_i(\varepsilon)))$ must behave uniformly (either the two of them fail, or the two of them does not fail). Then we know that $(\varepsilon_1 ; \rho_1(\varepsilon))$ is defined $\iff (\varepsilon_1 ; \rho_2(\varepsilon))$ is defined. Let us suppose that both $(\varepsilon_i ; \rho_i(\varepsilon))$ do not fail (otherwise the result holds immediately). Then we know that

$$\triangleright^{1+k} (n', \varepsilon_{f1} u_{1f}' :: \mathbf{G}_{12}', \varepsilon_{f2} u_{2f}' :: \mathbf{G}_{22}') \in \mathcal{V}_\rho[\mathbf{G}_2']$$

14. Then by the reduction rule of application terms, we obtain that:

$$\begin{aligned} (\varepsilon_{\lambda 1}(\lambda \mathbf{x} : \mathbf{G}_{x1}. \mathbf{t}_1) :: \mathbf{G}'_{11} \longrightarrow \mathbf{G}'_{12}) (\varepsilon'_1 u'_1 :: \mathbf{G}'_{11}) &\xrightarrow{1} \varepsilon_{z1} ([\varepsilon_1''' u'_1 :: \mathbf{G}_{x1}]/\mathbf{x}] \mathbf{t}_1) :: \mathbf{G}'_{12} \\ &\xrightarrow{k} \varepsilon_{z1} (\varepsilon_1'' u_{1f}' :: -) :: \mathbf{G}'_{12} \\ &\xrightarrow{0} \varepsilon_{f1} u_{1f}' :: \mathbf{G}'_{12} \end{aligned}$$

and

$$\begin{aligned} (\varepsilon_{\lambda 2}(\lambda \mathbf{x} : \mathbf{G}_{x2}. \mathbf{t}_2) :: \mathbf{G}'_{21} \longrightarrow \mathbf{G}'_{22}) (\varepsilon'_2 u'_2 :: \mathbf{G}'_{21}) &\xrightarrow{*} \varepsilon_{z2} ([\varepsilon_2''' u'_2 :: \mathbf{G}_{x2}]/\mathbf{x}] \mathbf{t}_2) :: \mathbf{G}'_{22} \\ &\xrightarrow{*} \varepsilon_{z2} (\varepsilon_2'' u_{2f}' :: -) :: \mathbf{G}'_{22} \\ &\xrightarrow{*} \varepsilon_{f2} u_{2f}' :: \mathbf{G}'_{22} \end{aligned}$$

then we have to prove that

$$\triangleright^{1+k} (n', \varepsilon_{f1} u_{1f}' :: \mathbf{G}'_{12}, \varepsilon_{f2} u_{2f}' :: \mathbf{G}'_{22}') \in \mathcal{V}_\rho[\mathbf{G}_2']$$

as we have already shown.

Case ($\mathbf{G} = \forall \mathbf{X}. \mathbf{G}_1''$ and $\mathbf{G}' = \forall \mathbf{X}. \mathbf{G}'_1$). We know that

1. $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}_1''$, and $\mathbf{G}' = \forall \mathbf{X}. \mathbf{G}'_1$ for some \mathbf{G}_1'' and \mathbf{G}'_1 .
2. We know that $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\forall \mathbf{X}. \mathbf{G}_1'']$, where $\mathbf{v}_i = \varepsilon_i(\Lambda \mathbf{X}. \mathbf{t}_i) :: \forall \mathbf{X}. \mathbf{G}_{i1}''$, for $\mathbf{G}_{i1}'' = \rho_i(\mathbf{G}_1'')$, and $\varepsilon_i : \mathbf{G}_i \sim \forall \mathbf{X}. \mathbf{G}_{i1}''$, for some \mathbf{G}_i .
3. We have to prove that:

$$(n, \varepsilon_1^\rho \mathbf{v}_1 :: \forall \mathbf{X}. \mathbf{G}'_{11}, \varepsilon_2^\rho \mathbf{v}_2 :: \forall \mathbf{X}. \mathbf{G}'_{11}) \in \mathcal{J}_\rho[\forall \mathbf{X}. \mathbf{G}'_1]$$

for $\varepsilon_i^\rho = \rho_i(\varepsilon)$, and $\mathbf{G}'_{i1} = \rho_i(\mathbf{G}'_1)$.

4. By Lemma B.40 and B.51, we know that $\varepsilon_i = \forall \mathbf{X}. \text{schm}(\varepsilon_i)$ and $\varepsilon_i^\rho = \forall \mathbf{X}. \text{schm}(\varepsilon_i^\rho)$.
5. As $\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho = \forall \mathbf{X}. \text{schm}(\varepsilon_i) \mathbin{\text{;}} \forall \mathbf{X}. \text{schm}(\varepsilon_i^\rho) = \forall \mathbf{X}. (\text{schm}(\varepsilon_i) \mathbin{\text{;}} \text{schm}(\varepsilon_i^\rho)) = \forall \mathbf{X}. \text{schm}(\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho)$ (by Lemma B.49), this means that proving that

$$(\varepsilon_1 \mathbin{\text{;}} \varepsilon_1^\rho) \text{ is defined} \iff (\varepsilon_2 \mathbin{\text{;}} \varepsilon_2^\rho) \text{ is defined}$$

is equivalent to prove that

$$(\text{schm}(\varepsilon_1) \mathbin{\text{;}} \text{schm}(\varepsilon_1^\rho)) \text{ is defined} \iff (\text{schm}(\varepsilon_2) \mathbin{\text{;}} \text{schm}(\varepsilon_2^\rho)) \text{ is defined}$$

6. Suppose that $\varepsilon_i \mathbin{\text{;}} \varepsilon_i^\rho = \varepsilon_{\Lambda i}$. Then we have to prove that

$$(n, \varepsilon_{\Lambda 1}(\Lambda \mathbf{X}. \mathbf{t}_1) :: \forall \mathbf{X}. \mathbf{G}'_{11}, \varepsilon_{\Lambda 2}(\Lambda \mathbf{X}. \mathbf{t}_2) :: \forall \mathbf{X}. \mathbf{G}'_{21}) \in \mathcal{V}_\rho[\forall \mathbf{X}. \mathbf{G}'_1]$$

i.e. let us take any $\mathbf{F}_1, \mathbf{F}_2, R \in \text{REL}[\mathbf{F}_1, \mathbf{F}_2]$, then we have to prove that

$$(n, \varepsilon_{\Lambda 1}(\Lambda \mathbf{X}. \mathbf{t}_1) :: \forall \mathbf{X}. \mathbf{G}'_{11} [\mathbf{F}_1], \varepsilon_{\Lambda 2}(\Lambda \mathbf{X}. \mathbf{t}_2) :: \forall \mathbf{X}. \mathbf{G}'_{21} [\mathbf{F}_2]) \in \mathcal{J}_{\rho'}[\mathbf{G}'_1]$$

for $\rho' = (\rho; \mathbf{X} \mapsto (\mathbf{F}_1, \mathbf{F}_2, R))$

7. By the definition of $\mathcal{V}_\rho[\forall \mathbf{X}. \mathbf{G}_1'']$, we know that for all $\mathbf{F}'_1, \mathbf{F}'_2, R' \in \text{REL}[\mathbf{F}'_1, \mathbf{F}'_2]$

$$(n, \varepsilon_1(\Lambda \mathbf{X}. \mathbf{t}_1) :: \forall \mathbf{X}. \mathbf{G}_{11}'' [\mathbf{F}'_1], \varepsilon_2(\Lambda \mathbf{X}. \mathbf{t}_2) :: \forall \mathbf{X}. \mathbf{G}_{21}'' [\mathbf{F}'_2]) \in \mathcal{J}_{(\rho; \mathbf{X} \mapsto (\mathbf{F}'_1, \mathbf{F}'_2, R'))}[\mathbf{G}_1'']$$

We instantiate this with $\mathbf{F}_1, \mathbf{F}_2, R \in \text{REL}[\mathbf{F}_1, \mathbf{F}_2]$ to derive that

$$(n, \varepsilon_1(\Lambda \mathbf{X}. \mathbf{t}_1) :: \forall \mathbf{X}. \mathbf{G}_{11}'' [\mathbf{F}_1], \varepsilon_2(\Lambda \mathbf{X}. \mathbf{t}_2) :: \forall \mathbf{X}. \mathbf{G}_{21}'' [\mathbf{F}_2]) \in \mathcal{J}_{\rho'}[\mathbf{G}_1'']$$

8. By the reduction rules, we know that

$$(\varepsilon_i(\Lambda \mathbf{X}. \mathbf{t}_i) :: \forall \mathbf{X}. \mathbf{G}_{i1}'' [\mathbf{F}_i]) \xrightarrow{1} (\text{schm}(\varepsilon_i[\mathbf{X} \mapsto \mathbf{F}_i]) (\mathbf{t}_i[\mathbf{X} \mapsto \mathbf{F}_i]) :: \mathbf{G}_{i1}'' [\mathbf{X}/\mathbf{F}_i])$$

We know that both terms fail an the result holds, or if

$$(\text{schm}(\varepsilon_1[\mathbf{X} \mapsto \mathbf{F}_1]) (\mathbf{t}_1[\mathbf{X} \mapsto \mathbf{F}_1]) :: \mathbf{G}_{11}'' [\mathbf{X}/\mathbf{F}_1]) \longrightarrow^k \mathbf{v}_{1f}$$

then

$$(\text{schm}(\varepsilon_2[\mathbf{X} \mapsto \mathbf{F}_2]) (\mathbf{t}_2[\mathbf{X} \mapsto \mathbf{F}_2]) :: \mathbf{G}_{21}'' [\mathbf{X}/\mathbf{F}_2]) \longrightarrow^* \mathbf{v}_{2f}$$

and $\triangleright^{1+k}(n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho'}[\mathbf{G}_1'']$.

9. We know by Lemmas B.51 and B.49 that

$$\begin{aligned} \text{schm}(\varepsilon_{\Lambda_i})[\mathbf{X} \mapsto \mathbf{F}_i] &= \text{schm}(\varepsilon_i \wp \varepsilon_i^\rho)[\mathbf{X} \mapsto \mathbf{F}_i] = (\text{schm}(\varepsilon_i) \wp \text{schm}(\varepsilon_i^\rho))[\mathbf{X} \mapsto \mathbf{F}_i] = \\ &(\text{schm}(\varepsilon_i)[\mathbf{X} \mapsto \mathbf{F}_i]) \wp (\text{schm}(\varepsilon_i^\rho)[\mathbf{X} \mapsto \mathbf{F}_i]) = (\text{schm}(\varepsilon_i)[\mathbf{X} \mapsto \mathbf{F}_i]) \wp (\text{schm}(\varepsilon_i^{\rho'})) \end{aligned}$$

where for $\varepsilon_i^{\rho'} = \rho'_i(\varepsilon)$.

10. We apply induction hypothesis to the previous result with $(n-1-k, \mathbf{G}'_1, \mathbf{G}'_1)$ and $\triangleright^{1+k}(n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho'}[\mathbf{G}'_1]$, $\text{schm}(\varepsilon) : \mathbf{G}'_1 \sim \mathbf{G}'_1$, to obtain that:

$$\triangleright^{1+k}(n, (\text{schm}(\varepsilon_1^{\rho'}))\mathbf{v}_{1f} :: \mathbf{G}'_1, (\text{schm}(\varepsilon_2^{\rho'}))\mathbf{v}_{2f} :: \mathbf{G}'_1) \in \mathcal{J}_{\rho'}[\mathbf{G}'_1]$$

Then \mathbf{v}_{if} has to have the form: $\mathbf{v}_{if} = \varepsilon_{ci} \mathbf{u}_{if} :: \mathbf{G}'_{i1}$, where $\varepsilon''_1 \wp (\text{schm}(\varepsilon_1)[\mathbf{X} \mapsto \mathbf{F}_1]) = \varepsilon_{c1}$, and $\varepsilon''_2 \wp (\text{schm}(\varepsilon_2)[\mathbf{X} \mapsto \mathbf{F}_2]) = \varepsilon_{c2}$, for some $\varepsilon''_i, \mathbf{u}_{if}$. Suppose that $(\text{schm}(\varepsilon_1)[\mathbf{X} \mapsto \mathbf{F}_1]) \wp (\text{schm}(\varepsilon_1^{\rho'})) = \varepsilon_{z1}$ is defined for now. Then by Lemma B.18, we know that $(\varepsilon''_1 \wp (\text{schm}(\varepsilon_1)[\mathbf{X} \mapsto \mathbf{F}_1])) \wp (\text{schm}(\varepsilon_1^{\rho'})) = \varepsilon_{f1} \iff \varepsilon''_1 \wp ((\text{schm}(\varepsilon_1)[\mathbf{X} \mapsto \mathbf{F}_1]) \wp (\text{schm}(\varepsilon_1^{\rho'}))) = \varepsilon_{f1}$, then $(\text{schm}(\varepsilon_1)[\mathbf{X} \mapsto \mathbf{F}_1]) \wp (\text{schm}(\varepsilon_1^{\rho'}))$ must behave uniformly (either the two of them fail, or the two of them does not fail). Then we know that $(\varepsilon_1 \wp \varepsilon_1^\rho)$ is defined $\iff (\varepsilon_2 \wp \varepsilon_2^\rho)$ is defined. Let us suppose that both $(\varepsilon_i \wp \varepsilon_i^\rho)$ do not fail (otherwise the result holds immediately). Then we know that

$$\triangleright^{1+k}(n, \varepsilon_{f1} \mathbf{u}_{1f} :: \mathbf{G}'_{11}, \varepsilon_{f2} \mathbf{u}_{2f} :: \mathbf{G}'_{21}) \in \mathcal{V}_{\rho'}[\mathbf{G}'_1]$$

11. Then by the reduction rule of type application, we obtain that:

$$\begin{aligned} (\varepsilon_{\Lambda_1}(\mathbf{A}\mathbf{X}.t_1) :: \forall \mathbf{X}. \mathbf{G}'_{11}) [\mathbf{F}_1] &\xrightarrow{1} \varepsilon_{z1} (t_1[\mathbf{X} \mapsto \mathbf{F}_1]) :: \mathbf{G}'_{11}[\mathbf{F}_1/\mathbf{X}] \\ &\xrightarrow{k} \varepsilon_{z1} (\varepsilon''_1 \mathbf{u}_{1f} :: -) :: \mathbf{G}'_{11}[\mathbf{F}_1/\mathbf{X}] \\ &\xrightarrow{0} \varepsilon_{f1} \mathbf{u}_{1f} :: \mathbf{G}'_{11}[\mathbf{F}_1/\mathbf{X}] \end{aligned}$$

and

$$\begin{aligned} (\varepsilon_{\Lambda_2}(\mathbf{A}\mathbf{X}.t_2) :: \forall \mathbf{X}. \mathbf{G}'_{21}) [\mathbf{F}_2] &\xrightarrow{*} \varepsilon_{z2} (t_2[\mathbf{X} \mapsto \mathbf{F}_2]) :: \mathbf{G}'_{21}[\mathbf{F}_2/\mathbf{X}] \\ &\xrightarrow{*} \varepsilon_{z12} (\varepsilon''_2 \mathbf{u}_{2f} :: -) :: \mathbf{G}'_{21}[\mathbf{F}_2/\mathbf{X}] \\ &\xrightarrow{*} \varepsilon_{f2} \mathbf{u}_{2f} :: \mathbf{G}'_{21}[\mathbf{F}_2/\mathbf{X}] \end{aligned}$$

then we have to prove that

$$\triangleright^{1+k}(n, \varepsilon_{f1} \mathbf{u}_{1f} :: \mathbf{G}'_{11}, \varepsilon_{f2} \mathbf{u}_{2f} :: \mathbf{G}'_{22}) \in \mathcal{V}_{\rho'}[\mathbf{G}'_1]$$

as we have already shown.

Case $(\varepsilon : ?_\delta \sim \mathbf{B})$. We know that

1. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[?_\delta]$, we get that $(n, \mathbf{v}_1, \mathbf{v}_2) \in \text{ATOM}[\rho_1(?_\delta), \rho_2(?_\delta)]$, and therefore, $\mathbf{v}_i = \varepsilon_{ui} \mathbf{u}_i :: \rho_i(?_\delta)$ and $\varepsilon_{ui} : \mathbf{G}_{ui} \sim \rho_i(?_\delta)$, for some \mathbf{G}_{ui} .

2. We are required to prove that

$$(n, \rho_1(\varepsilon) \mathbf{v}_1 :: \mathbf{B}, \rho_2(\varepsilon) \mathbf{v}_2 :: \mathbf{B}) \in \mathcal{J}_\rho[\mathbf{B}]$$

3. Since $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \mathbf{B}$, by Lemma B.41, we know that $\forall \mathbf{S} \in \varepsilon$, $\mathbf{S} = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})]$, or $\mathbf{S} = [(\mathbf{B}, \text{inj}_{\mathbf{X}}, \mathbf{B})]$ and $\mathbf{X} : \mathbf{B} \in \delta$. Note that $\forall \mathbf{S} \in \varepsilon$, $[\mathbf{S}] : ?_\delta \sim \mathbf{B}$.
4. We first prove that for $\forall \mathbf{S} \in \varepsilon$, we get that $(n, \rho_1([\mathbf{S}]) \mathbf{v}_1 :: \rho_1(\mathbf{B}), \rho_2([\mathbf{S}]) \mathbf{v}_2 :: \rho_2(\mathbf{B})) \in \mathcal{T}_\rho[\mathbf{B}]$.
5. Let us suppose that $\mathbf{S} = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})]$. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[?_\delta]$, we instantiate this with $\mathbf{G}_{\mathbf{Ra}} = \mathbf{B}$, where $\varepsilon'_a = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})] = [\mathbf{S}]$, such that $\vdash \varepsilon'_a : \delta \rightarrow \mathbf{G}_{\mathbf{Ra}}$. Thus, we get that

$$(n, \rho_1(\varepsilon'_a) \mathbf{v}_1 :: \mathbf{B}, \rho_2(\varepsilon'_a) \mathbf{v}_2 :: \mathbf{B}) \in \mathcal{T}_\rho[\mathbf{B}]$$

where $\varepsilon'_a = \rho_i(\varepsilon'_a) = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})] = [\mathbf{S}]$. Or what is the same, we get that

$$(n, \rho_1([\mathbf{S}]) \mathbf{v}_1 :: \mathbf{B}, \rho_2([\mathbf{S}]) \mathbf{v}_2 :: \mathbf{B}) \in \mathcal{T}_\rho[\mathbf{B}]$$

By the definition of $\mathcal{T}_\rho[\mathbf{B}]$, if $\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon'_a) = \varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon)$ fails then $\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon'_a) = \varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon)$ also fails. Also, by the definition of $\mathcal{T}_\rho[\mathbf{B}]$, if $\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon'_a)$ succeeds then $\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon'_a)$ also succeeds and $(n, (\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon'_a)) \mathbf{u}_1 :: \mathbf{B}, (\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon'_a)) \mathbf{u}_2 :: \mathbf{B}) \in \mathcal{V}_\rho[\mathbf{B}]$.

6. Now, let us suppose that $\mathbf{S} = [(\mathbf{B}, \text{inj}_{\mathbf{X}}, \mathbf{B})]$ and $\mathbf{X} : \mathbf{B} \in \delta$. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[?_\delta]$, we instantiate this with $\mathbf{G}_{\mathbf{Rb}} = \mathbf{X} (\delta \vdash \mathbf{X})$, where $\varepsilon'_b = [(\mathbf{B}, \text{inj}_{\mathbf{X}}, \mathbf{B})] = [\mathbf{S}] (\mathbf{X} : \mathbf{B} \in \delta)$, such that $\vdash \varepsilon'_b : \delta \rightarrow \mathbf{G}_{\mathbf{Rb}}$. Thus, we get that

$$(n, \rho_1(\varepsilon'_b) \mathbf{v}_1 :: \mathbf{B}, \rho_2(\varepsilon'_b) \mathbf{v}_2 :: \mathbf{B}) \in \mathcal{T}_\rho[\mathbf{B}]$$

where $\varepsilon'_b = \rho_i(\varepsilon'_b) = [(\mathbf{B}, \text{inj}_{\mathbf{X}}, \mathbf{B})] = [\mathbf{S}]$. Or what is the same, we get that

$$(n, \rho_1([\mathbf{S}]) \mathbf{v}_1 :: \mathbf{B}, \rho_2([\mathbf{S}]) \mathbf{v}_2 :: \mathbf{B}) \in \mathcal{T}_\rho[\mathbf{B}]$$

By the definition of $\mathcal{T}_\rho[\mathbf{B}]$, if $\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon'_b) = \varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon)$ fails then $\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon'_b) = \varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon)$ also fails. Also, by the definition of $\mathcal{T}_\rho[\mathbf{B}]$, if $\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon'_b)$ succeeds then $\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon'_b)$ also succeeds and $(n, (\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon'_b)) \mathbf{u}_1 :: \mathbf{B}, (\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon'_b)) \mathbf{u}_2 :: \mathbf{B}) \in \mathcal{V}_\rho[\mathbf{B}]$.

7. We know that $\forall \mathbf{S} \in \varepsilon$, $\varepsilon_1 \mathbin{\varepsilon} \rho_1([\mathbf{S}]) \iff \varepsilon_2 \mathbin{\varepsilon} \rho_2([\mathbf{S}])$. Therefore, by Lemma B.52, we get that $(\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon))$ fails, then $(\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon))$ fails, and the result follows. Otherwise, we get that $(\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon))$ succeeds, then $(\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon))$ succeeds, and therefore, we are required to prove that

$$(n, (\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon)) \mathbf{u}_1 :: \mathbf{B}, (\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon)) \mathbf{u}_2 :: \mathbf{B}) \in \mathcal{V}_\rho[\mathbf{B}]$$

8. Since $\varepsilon_{\mathbf{ui}} : \mathbf{G}_{\mathbf{ui}} \sim \rho_i(?_\delta)$, for some $\mathbf{G}_{\mathbf{ui}}$, $\rho_i(\varepsilon) : \rho_i(?_\delta) \sim \mathbf{B}$ and $(\varepsilon_{\mathbf{ui}} \mathbin{\varepsilon} \rho_i(\varepsilon))$ is defined, by Lemma B.43, we know that $(\varepsilon_{\mathbf{ui}} \mathbin{\varepsilon} \rho_i(\varepsilon)) = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\} = \text{reflEv}(\mathbf{B})$. Since $\vdash \varepsilon_{\mathbf{ui}} \mathbf{u}_i :: \rho_i(?_\delta) : \rho_i(?_\delta)$, $\varepsilon_{\mathbf{ui}} : \mathbf{G}_{\mathbf{ui}} \sim \rho_i(?_\delta)$, $\rho_i(\varepsilon) : \rho_i(?_\delta) \sim \mathbf{B}$ and $\varepsilon_{\mathbf{ui}} \mathbin{\varepsilon} \rho_i(\varepsilon)$ is defined, then $\mathbf{G}_{\mathbf{ui}} = \mathbf{B}$ and $\mathbf{u}_i = \mathbf{b}$. Thus, we are required to prove that

$$(n, \text{reflEv}(\mathbf{B}) \mathbf{b} :: \mathbf{B}, \text{reflEv}(\mathbf{B}) \mathbf{b} :: \mathbf{B}) \in \mathcal{V}_\rho[\mathbf{B}]$$

which follows immediately. Thus, the result holds.

Case $(\varepsilon : ?_\delta \sim \mathbf{X})$. We know that

1. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[?_\delta]$, we get that $(n, \mathbf{v}_1, \mathbf{v}_2) \in \text{ATOM}[\rho_1(?_\delta), \rho_2(?_\delta)]$, and therefore, $\mathbf{v}_i = \varepsilon_{\mathbf{ui}} \mathbf{u}_i :: \rho_i(?_\delta)$ and $\varepsilon_{\mathbf{ui}} : \mathbf{G}_{\mathbf{ui}} \sim \rho_i(?_\delta)$, for some $\mathbf{G}_{\mathbf{ui}}$.
2. We are required to prove that

$$(n, \rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{X}), \rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{X})) \in \mathcal{T}_\rho[\mathbf{X}]$$

3. Since $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \mathbf{X}$, by Lemma B.41, we know that $\forall \mathbf{S} \in \varepsilon$, $\mathbf{S} = [(\mathbf{X}, \text{inj}_{\mathbf{X}}, \mathbf{X})]$ and $\mathbf{X} : \mathbf{X} \in \delta$, or $\mathbf{S} = [(\mathbf{X}, \text{inj}_{\mathbf{Y}}, \mathbf{X})]$, $\mathbf{X} \neq \mathbf{Y}$, $\mathbf{X} : \mathbf{X} \in \delta$, and $\mathbf{Y} : \mathbf{X} \in \delta$. Note that $\forall \mathbf{S} \in \varepsilon$, $[\mathbf{S}] : ?_\delta \sim \mathbf{X}$.

4. We first prove that for $\forall \mathbf{S} \in \varepsilon$, we get that $(n, \rho_1([\mathbf{S}]) \mathbf{v}_1 :: \rho_1(\mathbf{X}), \rho_2([\mathbf{S}]) \mathbf{v}_2 :: \rho_2(\mathbf{X})) \in \mathcal{T}_\rho[\mathbf{X}]$.

5. Let us suppose that $\mathbf{S} = [(\mathbf{X}, \text{inj}_{\mathbf{X}}, \mathbf{X})]$ and $\mathbf{X} : \mathbf{X} \in \delta$. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[?_\delta]$, we instantiate this with $\mathbf{G}_{\mathbf{Ra}} = \mathbf{X}$, where $\varepsilon'_a = [(\mathbf{X}, \text{inj}_{\mathbf{X}}, \mathbf{X})] = [\mathbf{S}]$, such that $\vdash \varepsilon'_a : \delta \rightarrow \mathbf{G}_{\mathbf{Ra}}$. Thus, we get that

$$(n, \rho_1(\varepsilon'_a) \mathbf{v}_1 :: \rho_1(\mathbf{X}), \rho_2(\varepsilon'_a) \mathbf{v}_2 :: \rho_2(\mathbf{X})) \in \mathcal{T}_\rho[\mathbf{X}]$$

where $\rho_i(\varepsilon'_a) = \rho_i([(\mathbf{X}, \text{inj}_{\mathbf{X}}, \mathbf{X})]) = [(\rho_i(\mathbf{X}), \text{inj}_{\mathbf{X}}, \text{refl}_{\rho_i(\mathbf{X})}^\square)] = \rho_i([\mathbf{S}])$. Or what is the same, we get that

$$(n, \rho_1([\mathbf{S}]) \mathbf{v}_1 :: \rho_1(\mathbf{X}), \rho_2([\mathbf{S}]) \mathbf{v}_2 :: \rho_2(\mathbf{X})) \in \mathcal{T}_\rho[\mathbf{X}]$$

By the definition of $\mathcal{T}_\rho[\mathbf{X}]$, if $\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1([\mathbf{S}])$ fails then $\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2([\mathbf{S}])$ also fails. Also, by the definition of $\mathcal{T}_\rho[\mathbf{X}]$, if $\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1([\mathbf{S}])$ succeeds then $\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2([\mathbf{S}])$ also succeeds and

$$(n, (\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1([\mathbf{S}])) \mathbf{u}_1 :: \rho_1(\mathbf{X}), (\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2([\mathbf{S}])) \mathbf{u}_2 :: \rho_2(\mathbf{X})) \in \mathcal{V}_\rho[\mathbf{X}]$$

Since $\varepsilon_{\mathbf{u1}} : \mathbf{G}_{\mathbf{u1}} \sim \rho_i(?_\delta)$, for some $\mathbf{G}_{\mathbf{u1}}$, $\rho_i([\mathbf{S}]) : \rho_i(?_\delta) \sim \mathbf{B}_i$ and $(\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_i([\mathbf{S}]))$ is defined, by Lemma B.43, we know that $(\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_i([\mathbf{S}])) = [(\mathbf{B}_i, \mathbf{B}_i, \mathbf{B}_i)] = \text{reflEv}(\mathbf{B}_i)$. Since $\vdash \varepsilon_{\mathbf{u1}} \mathbf{u}_i :: \rho_i(?_\delta) : \rho_i(?_\delta)$, $\varepsilon_{\mathbf{u1}} : \mathbf{G}_{\mathbf{u1}} \sim \rho_i(?_\delta)$, $\rho_i([\mathbf{S}]) : \rho_i(?_\delta) \sim \mathbf{B}_i$ and $\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_i([\mathbf{S}])$ is defined, then $\mathbf{G}_{\mathbf{u1}} = \mathbf{B}_i$ and $\mathbf{u}_i = \mathbf{b}_i$. Thus, we know that

$$(n, \text{reflEv}(\mathbf{B}_1) \mathbf{b}_1 :: \mathbf{B}_1, \text{reflEv}(\mathbf{B}_2) \mathbf{b}_2 :: \mathbf{B}_2) \in \mathcal{V}_\rho[\mathbf{X}]$$

6. Now, let us suppose that $\mathbf{S} = [(\mathbf{X}, \text{inj}_{\mathbf{Y}}, \mathbf{X})]$, $\mathbf{X} \neq \mathbf{Y}$, $\mathbf{X} : \mathbf{X} \in \delta$ and $\mathbf{Y} : \mathbf{X} \in \delta$. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[?_\delta]$, we instantiate this with $\mathbf{G}_{\mathbf{Rb}} = \mathbf{Y}$ ($\delta \vdash \mathbf{Y}$), where $\varepsilon'_b = [(\mathbf{X}, \text{inj}_{\mathbf{Y}}, \mathbf{X})] = [\mathbf{S}]$ ($\mathbf{Y} : \mathbf{X} \in \delta$), such that $\vdash \varepsilon'_b : \delta \rightarrow \mathbf{G}_{\mathbf{Rb}}$. Thus, we get that

$$(n, \rho_1(\varepsilon'_b) \mathbf{v}_1 :: \rho_1(\mathbf{X}), \rho_2(\varepsilon'_b) \mathbf{v}_2 :: \rho_2(\mathbf{X})) \in \mathcal{T}_\rho[\mathbf{X}]$$

where $\rho_i(\varepsilon'_b) = \rho_i([(\mathbf{X}, \text{inj}_{\mathbf{Y}}, \mathbf{X})]) = [(\rho_i(\mathbf{X}), \text{inj}_{\mathbf{Y}}, \text{refl}_{\rho_i(\mathbf{X})}^\square)] = [\mathbf{S}]$. Or what is the same, we get that

$$(n, \rho_1([\mathbf{S}]) \mathbf{v}_1 :: \rho_1(\mathbf{X}), \rho_2([\mathbf{S}]) \mathbf{v}_2 :: \rho_2(\mathbf{X})) \in \mathcal{T}_\rho[\mathbf{X}]$$

By the definition of $\mathcal{T}_\rho[\mathbf{X}]$, if $\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1([\mathbf{S}])$ fails then $\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2([\mathbf{S}])$ also fails. Also, by the definition of $\mathcal{T}_\rho[\mathbf{X}]$, if $\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1([\mathbf{S}])$ succeeds then $\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2([\mathbf{S}])$ also succeeds and

$$(n, (\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1([\mathbf{S}])) \mathbf{u}_1 :: \rho_1(\mathbf{X}), (\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2([\mathbf{S}])) \mathbf{u}_2 :: \rho_2(\mathbf{X})) \in \mathcal{V}_\rho[\mathbf{X}]$$

We know that $\rho_i(\mathbf{X}) = \mathbf{B}_i$. Since $\varepsilon_{\mathbf{u1}} : \mathbf{G}_{\mathbf{u1}} \sim \rho_i(?_\delta)$, for some $\mathbf{G}_{\mathbf{u1}}$, $\rho_i([\mathbf{S}]) : \rho_i(?_\delta) \sim \mathbf{B}_i$ and $(\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_i([\mathbf{S}]))$ is defined, by Lemma B.43, we know that $(\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_i([\mathbf{S}])) = [(\mathbf{B}_i, \mathbf{B}_i, \mathbf{B}_i)] = \text{reflEv}(\mathbf{B}_i)$. Since $\vdash \varepsilon_{\mathbf{u1}} \mathbf{u}_i :: \rho_i(?_\delta) : \rho_i(?_\delta)$, $\varepsilon_{\mathbf{u1}} : \mathbf{G}_{\mathbf{u1}} \sim \rho_i(?_\delta)$, $\rho_i([\mathbf{S}]) : \rho_i(?_\delta) \sim \mathbf{B}_i$ and $\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_i([\mathbf{S}])$ is defined, then $\mathbf{G}_{\mathbf{u1}} = \mathbf{B}_i$ and $\mathbf{u}_i = \mathbf{b}_i$. Thus, we know that

$$(n, \text{reflEv}(\mathbf{B}_1) \mathbf{b}_1 :: \mathbf{B}_1, \text{reflEv}(\mathbf{B}_2) \mathbf{b}_2 :: \mathbf{B}_2) \in \mathcal{V}_\rho[\mathbf{X}]$$

7. We know that $\forall \mathbf{S} \in \varepsilon$, $\varepsilon_1 \mathbin{\varepsilon} \rho_1([\mathbf{S}]) \iff \varepsilon_2 \mathbin{\varepsilon} \rho_2([\mathbf{S}])$. Therefore, by Lemma B.52, we get that $(\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon))$ fails, then $(\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon))$ fails, and the result follows. Otherwise, we get that $(\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon))$ succeeds, then $(\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon))$ succeeds, and therefore, we are required to prove that

$$\triangleright^j (n, (\varepsilon_{\mathbf{u1}} \mathbin{\varepsilon} \rho_1(\varepsilon)) \mathbf{u}_1 :: \rho_1(\mathbf{X}), (\varepsilon_{\mathbf{u2}} \mathbin{\varepsilon} \rho_2(\varepsilon)) \mathbf{u}_2 :: \rho_2(\mathbf{X})) \in \mathcal{V}_\rho[\mathbf{X}]$$

8. We know that $\rho_i(\mathbf{X}) = \mathbf{B}_i$. Since $\varepsilon_{\mathbf{ui}} : \mathbf{G}_{\mathbf{ui}} \sim \rho_i(?_\delta)$, for some $\mathbf{G}_{\mathbf{ui}}$, $\rho_i(\varepsilon) : \rho_i(?_\delta) \sim \mathbf{B}_i$ and $(\varepsilon_{\mathbf{ui}} \circ \rho_i(\varepsilon))$ is defined, by Lemma B.43, we know that $(\varepsilon_{\mathbf{ui}} \circ \rho_i(\varepsilon)) = [(\mathbf{B}_i, \mathbf{B}_i, \mathbf{B}_i)] = \text{reflEv}(\mathbf{B}_i)$. Since $\vdash \varepsilon_{\mathbf{ui}} \mathbf{u}_i :: \rho_i(?_\delta) : \rho_i(?_\delta)$, $\varepsilon_{\mathbf{ui}} : \mathbf{G}_{\mathbf{ui}} \sim \rho_i(?_\delta)$, $\rho_i(\varepsilon) : \rho_i(?_\delta) \sim \mathbf{B}_i$ and $\varepsilon_{\mathbf{ui}} \circ \rho_i(\varepsilon)$ is defined, then $\mathbf{G}_{\mathbf{ui}} = \mathbf{B}_i$ and $\mathbf{u}_i = \mathbf{b}_i$. Thus, we are required to prove that

$$(n, \text{reflEv}(\mathbf{B}_1) \mathbf{b}_1 :: \mathbf{B}_1, \text{reflEv}(\mathbf{B}_2) \mathbf{b}_2 :: \mathbf{B}_2) \in \mathcal{V}_\rho[\mathbf{X}]$$

as we have already shown.

Case $(\varepsilon : ?_\delta \sim \mathbf{G}' \text{ and } \mathbf{G}' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2, \text{ or } \mathbf{G}' = \mathbf{G}'_1 \times \mathbf{G}'_2 \text{ or } \mathbf{G}' = \forall \mathbf{X}. \mathbf{G}'_1)$. We know that

1. $\mathbf{G} = ?_\delta$, and $\mathbf{G}' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$, or $\mathbf{G}' = \mathbf{G}'_1 \times \mathbf{G}'_2$ or $\mathbf{G}' = \forall \mathbf{X}. \mathbf{G}'_1$.
2. Suppose $\mathbf{v}_i = \varepsilon_{\mathbf{ui}} \mathbf{u}_i :: \rho_i(?_\delta)$.
3. Since $\varepsilon : ?_\delta \sim \mathbf{G}'$, where $\mathbf{G}' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$, or $\mathbf{G}' = \mathbf{G}'_1 \times \mathbf{G}'_2$ or $\mathbf{G}' = \forall \mathbf{X}. \mathbf{G}'_1$, by Lemma B.41, we know that evidence ε can be decomposed in two evidence, ε_a , such that $\vdash \varepsilon_a : \delta \rightarrow \mathbf{G}_R$, $\varepsilon_b = \text{extract}_{c1}(\varepsilon)$, where $\varepsilon = \varepsilon_a \circ \varepsilon_b$, $\varepsilon_a : ?_\delta \sim \mathbf{G}_R$, $\varepsilon_b : \mathbf{G}_R \sim \mathbf{G}'$, $\mathcal{S}(?_\delta, \mathbf{G}') = \mathbf{G}_R$ and $c(\mathbf{G}') = c$.
4. Since $\varepsilon = \varepsilon_a \circ \varepsilon_b$, by Lemma B.51, we know that $\rho_i(\varepsilon) = \rho_i(\varepsilon_a) \circ \rho_i(\varepsilon_b)$.
5. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[?_\delta]$, we instantiate this with \mathbf{G}_R and ε_a , then we get that

$$(n, \rho_1(\varepsilon_a) \mathbf{v}_1 :: \rho_1(\mathbf{G}_R), \rho_2(\varepsilon_a) \mathbf{v}_2 :: \rho_2(\mathbf{G}_R)) \in \mathcal{J}_\rho[\mathbf{G}_R]$$

6. Since $\varepsilon_{\mathbf{ui}} \circ \rho_i(\varepsilon) = \varepsilon_{\mathbf{ui}} \circ (\rho_i(\varepsilon_a) \circ \rho_i(\varepsilon_b))$, by Lemma B.18, we know that $\varepsilon_{\mathbf{ui}} \circ \rho_i(\varepsilon) = \varepsilon_{\mathbf{ui}} \circ (\rho_i(\varepsilon_a) \circ \rho_i(\varepsilon_b)) = (\varepsilon_{\mathbf{ui}} \circ \rho_i(\varepsilon_a)) \circ \rho_i(\varepsilon_b)$. Therefore, we know that if $(\varepsilon_{\mathbf{ui}} \circ \rho_i(\varepsilon_a))$ fail, then $\varepsilon_{\mathbf{ui}} \circ \rho_i(\varepsilon)$ also fail.
7. By definition $\mathcal{J}_\rho[\mathbf{G}_R]$, we know that if $\varepsilon_{\mathbf{u1}} \circ \rho_1(\varepsilon_a)$ fails then $\varepsilon_{\mathbf{u2}} \circ \rho_2(\varepsilon_a)$ also fails. Thus, the result holds.
8. By the definition of $\mathcal{J}_\rho[\mathbf{G}_R]$, if $\varepsilon_{\mathbf{u1}} \circ \rho_1(\varepsilon_a)$ succeeds then $\varepsilon_{\mathbf{u2}} \circ \rho_2(\varepsilon_a)$ also succeeds and

$$(n, (\varepsilon_{\mathbf{u1}} \circ \rho_1(\varepsilon_a)) \mathbf{u}_1 :: \rho_1(\mathbf{G}_R), (\varepsilon_{\mathbf{u2}} \circ \rho_2(\varepsilon_a)) \mathbf{u}_2 :: \rho_2(\mathbf{G}_R)) \in \mathcal{V}_\rho[\mathbf{G}_R]$$

9. Since $(n, (\varepsilon_{\mathbf{u1}} \circ \rho_1(\varepsilon_a)) \mathbf{u}_1 :: \rho_1(\mathbf{G}_R), (\varepsilon_{\mathbf{u2}} \circ \rho_2(\varepsilon_a)) \mathbf{u}_2 :: \rho_2(\mathbf{G}_R)) \in \mathcal{V}_\rho[\mathbf{G}_R]$, $(n, \rho) \in \mathcal{D}[\Delta]$ and $\varepsilon_b : \mathbf{G}_R \sim \mathbf{G}'$, by the induction hypothesis on $(n, \mathbf{G}_R, \mathbf{G}')$, where $\mathbf{G}_R < ?_\delta$, we get that

$$(n, \rho_1(\varepsilon_b) ((\varepsilon_{\mathbf{u1}} \circ \rho_1(\varepsilon_a)) \mathbf{u}_1 :: \rho_1(\mathbf{G}_R)) :: \rho_1(\mathbf{G}'), \rho_2(\varepsilon_b) ((\varepsilon_{\mathbf{u2}} \circ \rho_2(\varepsilon_a)) \mathbf{u}_2 :: \rho_2(\mathbf{G}_R)) :: \rho_2(\mathbf{G}')) \in \mathcal{J}_\rho[\mathbf{G}']$$

By definition $\mathcal{J}_\rho[\mathbf{G}']$, we know that if $(\varepsilon_{\mathbf{u1}} \circ \rho_1(\varepsilon_a)) \circ \rho_1(\varepsilon_b)$ fails then $(\varepsilon_{\mathbf{u2}} \circ \rho_2(\varepsilon_a)) \circ \rho_2(\varepsilon_b)$ also fails. Thus, the result holds. By the definition of $\mathcal{J}_\rho[\mathbf{G}']$, if $(\varepsilon_{\mathbf{u1}} \circ \rho_1(\varepsilon_a)) \circ \rho_1(\varepsilon_b)$ succeeds then $(\varepsilon_{\mathbf{u2}} \circ \rho_2(\varepsilon_a)) \circ \rho_2(\varepsilon_b)$ also succeeds and

$$(n, ((\varepsilon_{\mathbf{u1}} \circ \rho_1(\varepsilon_a)) \circ \rho_1(\varepsilon_b)) \mathbf{u}_1 :: \rho(\mathbf{G}'), ((\varepsilon_{\mathbf{u2}} \circ \rho_2(\varepsilon_a)) \circ \rho_2(\varepsilon_b)) \mathbf{u}_2 :: \rho(\mathbf{G}')) \in \mathcal{V}_\rho[\mathbf{G}']$$

Thus, the result holds.

Case $(\varepsilon : \mathbf{G} \sim ?_\delta)$. We know that

1. $\mathbf{G}' = ?_\delta$.

2. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$, we know that $\mathbf{v}_i = \varepsilon_{\mathbf{u}_i} \mathbf{u}_i :: \rho_i(\mathbf{G})$.
3. We have to prove that $(n, \rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(?_\delta), \rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(?_\delta)) \in \mathcal{J}_\rho[?_\delta]$.
4. Therefore, we have to prove that if $\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)$ fails then $\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)$ fails, and that if $\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)$ succeeds then $\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)$ succeeds, and $(n, (\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)) \mathbf{u}_1 :: \rho_1(?_\delta), (\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)) \mathbf{u}_2 :: \rho_2(?_\delta)) \in \mathcal{V}_\rho[?_\delta]$.
5. Since $\varepsilon : \mathbf{G} \sim ?_\delta$, by Lemma B.61, we know that there exists \mathbf{G}_R^* and ε^* such that $\vdash \varepsilon^* : \delta \rightarrow \mathbf{G}_R$ and $\varepsilon \mathbin{\dot{;}} \varepsilon^*$ is defined. We know that $\varepsilon^* : ?_\delta \sim \mathbf{G}_R^*$, for some $\mathbf{G}_R^* < ?_\delta$ and $(\varepsilon \mathbin{\dot{;}} \varepsilon^*) : \mathbf{G} \sim \mathbf{G}_R^*$. Therefore, we can use the induction hypothesis on $(n, \mathbf{G}, \mathbf{G}_R^*)$, with $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$ and $(\varepsilon \mathbin{\dot{;}} \varepsilon^*) : \mathbf{G} \sim \mathbf{G}_R^*$, and we get that

$$(n, \rho_1(\varepsilon \mathbin{\dot{;}} \varepsilon^*) \mathbf{v}_1 :: \rho_1(\mathbf{G}_R^*), \rho_2(\varepsilon \mathbin{\dot{;}} \varepsilon^*) \mathbf{v}_2 :: \rho_2(\mathbf{G}_R^*)) \in \mathcal{J}_\rho[\mathbf{G}_R^*]$$

6. By Lemma B.51, we know that $\rho_i(\varepsilon \mathbin{\dot{;}} \varepsilon^*) = \rho_i(\varepsilon) \mathbin{\dot{;}} \rho_i(\varepsilon^*)$. By Lemma B.18, we know that $\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} (\rho_i(\varepsilon) \mathbin{\dot{;}} \rho_i(\varepsilon^*)) = (\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_i(\varepsilon)) \mathbin{\dot{;}} \rho_i(\varepsilon^*)$. By the definition of $\mathcal{J}_\rho[\mathbf{G}_R^*]$, we know that $\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} (\rho_1(\varepsilon) \mathbin{\dot{;}} \rho_1(\varepsilon^*)) = (\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)) \mathbin{\dot{;}} \rho_1(\varepsilon^*)$ fails then $\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} (\rho_2(\varepsilon) \mathbin{\dot{;}} \rho_2(\varepsilon^*)) = (\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)) \mathbin{\dot{;}} \rho_2(\varepsilon^*)$ also fails. Therefore, we get that if $\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)$ fails then $\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)$ fails, and the result holds. Also, we know that $\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} (\rho_1(\varepsilon) \mathbin{\dot{;}} \rho_1(\varepsilon^*)) = (\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)) \mathbin{\dot{;}} \rho_1(\varepsilon^*)$ succeeds then $\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} (\rho_2(\varepsilon) \mathbin{\dot{;}} \rho_2(\varepsilon^*)) = (\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)) \mathbin{\dot{;}} \rho_2(\varepsilon^*)$ also succeeds. Therefore, we get that if $\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)$ succeeds then $\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)$ succeeds, and thus, we are required to prove that $(n, (\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)) \mathbf{u}_1 :: \rho_1(?_\delta), (\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)) \mathbf{u}_2 :: \rho_2(?_\delta))$.
7. Thus, we are required to prove that \mathbf{G}_R and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$, then

$$(n, \rho_1(\varepsilon') ((\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)) \mathbf{u}_1 :: \rho_1(?_\delta)) :: \rho_1(\mathbf{G}_R), \rho_2(\varepsilon') ((\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)) \mathbf{u}_2 :: \rho_2(?_\delta)) :: \rho_2(\mathbf{G}_R)) \in \mathcal{J}_\rho[\mathbf{G}_R]$$

8. By Lemma B.18, $(\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_i(\varepsilon)) \mathbin{\dot{;}} \rho_i(\varepsilon') \iff \varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} (\rho_i(\varepsilon) \mathbin{\dot{;}} \rho_i(\varepsilon'))$ (either both $(\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_i(\varepsilon)) \mathbin{\dot{;}} \rho_i(\varepsilon')$ and $\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} (\rho_i(\varepsilon) \mathbin{\dot{;}} \rho_i(\varepsilon'))$, succeed or fail).
9. By Lemma B.51, $\rho_i(\varepsilon) \mathbin{\dot{;}} \rho_i(\varepsilon') = \rho_i(\varepsilon \mathbin{\dot{;}} \varepsilon')$. Therefore, $\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_i(\varepsilon \mathbin{\dot{;}} \varepsilon') = (\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_i(\varepsilon)) \mathbin{\dot{;}} \rho_i(\varepsilon')$.
10. We know that if $\varepsilon \mathbin{\dot{;}} \varepsilon'$ fails, then both $\rho_1(\varepsilon \mathbin{\dot{;}} \varepsilon')$ and $\rho_2(\varepsilon \mathbin{\dot{;}} \varepsilon')$ fail, and thus, both $(\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_i(\varepsilon)) \mathbin{\dot{;}} \rho_i(\varepsilon')$ and $(\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_i(\varepsilon)) \mathbin{\dot{;}} \rho_i(\varepsilon')$ fail; therefore the result holds.
11. Otherwise, $\varepsilon \mathbin{\dot{;}} \varepsilon'$ succeeds and $(\varepsilon \mathbin{\dot{;}} \varepsilon') : \mathbf{G} \sim \mathbf{G}_R$. Notice that $\delta \vdash \mathbf{G}_R$ and $\mathbf{G}_R < ?_\delta$. Thus, by the induction hypothesis on $(n, \mathbf{G}, \mathbf{G}_R)$, with $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$ and $(\varepsilon \mathbin{\dot{;}} \varepsilon') : \mathbf{G} \sim \mathbf{G}_R$, we get that

$$(n, \rho_1(\varepsilon \mathbin{\dot{;}} \varepsilon') \mathbf{v}_1 :: \rho_1(\mathbf{G}_R), \rho_2(\varepsilon \mathbin{\dot{;}} \varepsilon') \mathbf{v}_2 :: \rho_2(\mathbf{G}_R)) \in \mathcal{J}_\rho[\mathbf{G}_R]$$

12. By definition of $\mathcal{J}_\rho[\mathbf{G}_R]$, we know that

$$(\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon \mathbin{\dot{;}} \varepsilon')) \iff (\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon \mathbin{\dot{;}} \varepsilon'))$$

Therefore, we get that $(\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)) \mathbin{\dot{;}} \rho_1(\varepsilon') \iff (\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)) \mathbin{\dot{;}} \rho_2(\varepsilon')$. Thus, if $(\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)) \mathbin{\dot{;}} \rho_1(\varepsilon')$ fails, then $(\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)) \mathbin{\dot{;}} \rho_2(\varepsilon')$ also fails, and the result holds.

Otherwise, if $(\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon \mathbin{\dot{;}} \varepsilon')) = (\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)) \mathbin{\dot{;}} \rho_1(\varepsilon')$ succeeds then $(\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon \mathbin{\dot{;}} \varepsilon')) = (\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)) \mathbin{\dot{;}} \rho_2(\varepsilon')$ also succeeds, and

$$(n, (\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon \mathbin{\dot{;}} \varepsilon')) \mathbf{u}_1 :: \rho_1(\mathbf{G}_R), (\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon \mathbin{\dot{;}} \varepsilon')) \mathbf{u}_2 :: \rho_2(\mathbf{G}_R)) \in \mathcal{V}_\rho[\mathbf{G}_R]$$

Or what is the same, we get that

$$(n, ((\varepsilon_{\mathbf{u}_1} \mathbin{\dot{;}} \rho_1(\varepsilon)) \mathbin{\dot{;}} \rho_1(\varepsilon')) \mathbf{u}_1 :: \rho_1(\mathbf{G}_R), ((\varepsilon_{\mathbf{u}_2} \mathbin{\dot{;}} \rho_2(\varepsilon)) \mathbin{\dot{;}} \rho_2(\varepsilon')) \mathbf{u}_2 :: \rho_2(\mathbf{G}_R)) \in \mathcal{V}_\rho[\mathbf{G}_R]$$

Thus, the result holds.

□

Lemma B.40 Consider $v = \varepsilon \mathbf{u} :: \mathbf{G}$ and $\vdash \mathbf{v} : \mathbf{G}$.

1. If $\mathbf{G} = \mathbf{B}$ then $\varepsilon = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\}$.
2. If $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$ then $\varepsilon = \varepsilon_1 \rightarrow \varepsilon_2$, for $\varepsilon_1 = \text{dom}^{-1}(\varepsilon)$ and $\varepsilon_2 = \text{cod}(\varepsilon)$.
3. If $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$ then $\varepsilon = \varepsilon_1 \times \varepsilon_2$, for $\varepsilon_1 = \pi_1(\varepsilon)$ and $\varepsilon_2 = \pi_2(\varepsilon)$.
4. If $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}'$ then $\varepsilon = \forall \mathbf{X}. \varepsilon'$, for $\varepsilon' = \text{schm}(\varepsilon)$.

PROOF. Directly by Lemma B.41. □

Lemma B.41 Consider $\varepsilon : \mathbf{G} \sim \mathbf{G}'$.

1. If $\mathbf{G} = \mathbf{B}$ and $\mathbf{G}' = \mathbf{B}$, then $\varepsilon = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\}$.
2. If $\mathbf{G} = \mathbf{X}$ and $\mathbf{G}' = \mathbf{X}$, then $\varepsilon = \{(\mathbf{X}, \mathbf{X}, \mathbf{X})\}$.
3. If $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\mathbf{G}' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$, then $\varepsilon = \varepsilon_1 \rightarrow \varepsilon_2$, where $\varepsilon_1 = \text{dom}^{-1}(\varepsilon)$ and $\varepsilon_2 = \text{cod}(\varepsilon)$.
4. If $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$ and $\mathbf{G}' = \mathbf{G}'_1 \times \mathbf{G}'_2$, then $\varepsilon = \varepsilon_1 \times \varepsilon_2$, where $\varepsilon_1 = \pi_1(\varepsilon)$ and $\varepsilon_2 = \pi_2(\varepsilon)$.
5. If $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}_1$ and $\mathbf{G}' = \forall \mathbf{X}. \mathbf{G}'_1$, then $\varepsilon = \forall \mathbf{X}. \varepsilon'$, where $\varepsilon' = \text{schm}(\varepsilon)$.
6. If $\mathbf{G} = \mathbf{B}$ and $\mathbf{G}' = ?_\delta$, then $\forall \mathbf{S} \in \varepsilon$, $\mathbf{S} = (\mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{B}})$, or $\mathbf{S} = (\mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{X}})$ and $\mathbf{X} : \mathbf{B} \in \delta$.
7. If $\mathbf{G} = \mathbf{X}$ and $\mathbf{G}' = ?_\delta$, then $\forall \mathbf{S} \in \varepsilon$, $\mathbf{S} = (\mathbf{X}, \mathbf{X}, \text{inj}_{\mathbf{X}})$ and $\mathbf{X} : \mathbf{X} \in \delta$, or $\mathbf{S} = (\mathbf{X}, \mathbf{X}, \text{inj}_{\mathbf{Y}})$, $\mathbf{X} \neq \mathbf{Y}$, $\mathbf{X} : \mathbf{X} \in \delta$, and $\mathbf{Y} : \mathbf{X} \in \delta$.
8. If $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\mathbf{G}' = ?_\delta$, then $\varepsilon = \text{inj}_{\rightarrow 2}(\varepsilon_1) = \varepsilon_1 \ddagger \varepsilon_2$, where $\varepsilon_1 = \text{extract}_{\rightarrow 2}(\varepsilon)$ and $\varepsilon_2 = \{(?_\delta \rightarrow ?_\delta, ?_\delta \rightarrow ?_\delta, \text{inj}_{\rightarrow} (?_\delta \rightarrow ?_\delta))\}$.
9. If $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$ and $\mathbf{G}' = ?_\delta$, then $\varepsilon = \text{inj}_{\times 2}(\varepsilon_1) = \varepsilon_1 \ddagger \varepsilon_2$, where $\varepsilon_1 = \text{extract}_{\times 2}(\varepsilon)$ and $\varepsilon_2 = \{(?_\delta \times ?_\delta, ?_\delta \times ?_\delta, \text{inj}_{\times} (?_\delta \times ?_\delta))\}$.
10. If $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}_1$ and $\mathbf{G}' = ?_\delta$, then $\varepsilon = \text{inj}_{\forall 2}(\varepsilon_1) = \varepsilon_1 \ddagger \varepsilon_2$, where $\varepsilon_1 = \text{extract}_{\forall 2}(\varepsilon)$ and $\varepsilon_2 = \{(\forall \mathbf{X}. ?_\delta, \mathbf{X}, \forall \mathbf{X}. ?_\delta, \mathbf{X}, \text{inj}_{\forall} (\forall \mathbf{X}. ?_\delta, \mathbf{X}))\}$.
11. If $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \mathbf{B}$, then $\forall \mathbf{S} \in \varepsilon$, $\mathbf{S} = (\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})$, or $\mathbf{S} = (\mathbf{B}, \text{inj}_{\mathbf{X}}, \mathbf{B})$ and $\mathbf{X} : \mathbf{B} \in \delta$.
12. If $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \mathbf{X}$, then $\forall \mathbf{S} \in \varepsilon$, $\mathbf{S} = (\mathbf{X}, \text{inj}_{\mathbf{X}}, \mathbf{X})$ and $\mathbf{X} : \mathbf{X} \in \delta$, or $\mathbf{S} = (\mathbf{X}, \text{inj}_{\mathbf{Y}}, \mathbf{X})$, $\mathbf{X} \neq \mathbf{Y}$, $\mathbf{X} : \mathbf{X} \in \delta$, and $\mathbf{Y} : \mathbf{X} \in \delta$.
13. If $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$, then $\varepsilon = \text{inj}_{\rightarrow 1}(\varepsilon_1) = \varepsilon_2 \ddagger \varepsilon_1$, where $\varepsilon_1 = \text{extract}_{\rightarrow 1}(\varepsilon)$ and $\vdash \varepsilon_2 : \delta \rightarrow ?_\delta \rightarrow ?_\delta$.
14. If $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \mathbf{G}'_1 \times \mathbf{G}'_2$, then $\varepsilon = \text{inj}_{\times 1}(\varepsilon_1) = \varepsilon_2 \ddagger \varepsilon_1$, where $\varepsilon_1 = \text{extract}_{\times 1}(\varepsilon)$ and $\vdash \varepsilon_2 : \delta \rightarrow ?_\delta \times ?_\delta$.
15. If $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \forall \mathbf{X}. \mathbf{G}'_1$, then $\varepsilon = \text{inj}_{\forall 1}(\varepsilon_1) = \varepsilon_2 \ddagger \varepsilon_1$, where $\varepsilon_1 = \text{extract}_{\forall 1}(\varepsilon)$ and $\vdash \varepsilon_2 : \delta \rightarrow \forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}$.

PROOF. We know that $\varepsilon : \mathbf{G} \sim \mathbf{G}'$.

1. If $\mathbf{G} = \mathbf{B}$ and $\mathbf{G}' = \mathbf{B}$, then $\varepsilon = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\}$. Let us suppose that $(\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon$. We know that $\mathbf{c}_1 : \mathbf{G}'' \sqsubseteq \mathbf{B}$ and $\mathbf{c}_2 : \mathbf{G}'' \sqsubseteq \mathbf{B}$. Therefore, by Lemma B.11, we get that $\mathbf{G}'' = \mathbf{B}$, $\mathbf{c}_1 = \mathbf{B}$ and $\mathbf{c}_2 = \mathbf{B}$. Thus, the result holds.
2. If $\mathbf{G} = \mathbf{X}$ and $\mathbf{G}' = \mathbf{X}$, then $\varepsilon = \{(\mathbf{X}, \mathbf{X}, \mathbf{X})\}$. Let us suppose that $(\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon$. We know that $\mathbf{c}_1 : \mathbf{G}'' \sqsubseteq \mathbf{X}$ and $\mathbf{c}_2 : \mathbf{G}'' \sqsubseteq \mathbf{X}$. Therefore, by Lemma B.11, we get that $\mathbf{G}'' = \mathbf{X}$, $\mathbf{c}_1 = \mathbf{X}$ and $\mathbf{c}_2 = \mathbf{X}$. Thus, the result holds.
3. If $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\mathbf{G}' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$, then $\varepsilon = \varepsilon_1 \rightarrow \varepsilon_2$, where $\varepsilon_1 = \text{dom}^{-1}(\varepsilon)$ and $\varepsilon_2 = \text{cod}(\varepsilon)$. Let us suppose that $\mathbf{S} = (\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon$. We know that $\mathbf{c}_1 : \mathbf{G}'' \sqsubseteq \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\mathbf{c}_2 : \mathbf{G}'' \sqsubseteq \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$. Therefore, by Lemma B.11, we get that $\mathbf{G}'' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$, $\mathbf{c}_1 = \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}$ and $\mathbf{c}_2 = \mathbf{c}_{21} \rightarrow \mathbf{c}_{22}$. Thus, we know that

$$\begin{aligned} & \bullet \{\mathbf{S}_1\} = \text{dom}^{-1}(\mathbf{S}) = \{(\mathbf{G}'_1, \mathbf{c}_{11}, \mathbf{c}_{21})\} \\ & \bullet \{\mathbf{S}_2\} = \text{cod}(\mathbf{S}) = \{(\mathbf{G}'_2, \mathbf{c}_{12}, \mathbf{c}_{22})\} \\ & \bullet \{\mathbf{S}_1\} \rightarrow \{\mathbf{S}_2\} = \{(\mathbf{G}'_1 \rightarrow \mathbf{G}'_2, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22})\} = (\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) = \mathbf{S} \end{aligned}$$

Thus, the result holds.

4. If $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$ and $\mathbf{G}' = \mathbf{G}'_1 \times \mathbf{G}'_2$, then $\varepsilon = \varepsilon_1 \times \varepsilon_2$, where $\varepsilon_1 = \pi_1(\varepsilon)$ and $\varepsilon_2 = \pi_2(\varepsilon)$. Let us suppose that $\mathbf{S} = (\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon$. We know that $\mathbf{c}_1 : \mathbf{G}'' \sqsubseteq \mathbf{G}_1 \times \mathbf{G}_2$ and $\mathbf{c}_2 : \mathbf{G}'' \sqsubseteq \mathbf{G}'_1 \times \mathbf{G}'_2$. Therefore, by Lemma B.11, we get that $\mathbf{G}'' = \mathbf{G}'_1 \times \mathbf{G}'_2$, $\mathbf{c}_1 = \mathbf{c}_{11} \times \mathbf{c}_{12}$ and $\mathbf{c}_2 = \mathbf{c}_{21} \times \mathbf{c}_{22}$. Thus, we know that

$$\begin{aligned} & \bullet \{\mathbf{S}_1\} = \pi_1(\mathbf{S}) = \{(\mathbf{G}'_1, \mathbf{c}_{11}, \mathbf{c}_{21})\} \\ & \bullet \{\mathbf{S}_2\} = \pi_2(\mathbf{S}) = \{(\mathbf{G}'_2, \mathbf{c}_{12}, \mathbf{c}_{22})\} \\ & \bullet \{\mathbf{S}_1\} \times \{\mathbf{S}_2\} = \{(\mathbf{G}'_1 \times \mathbf{G}'_2, \mathbf{c}_{11} \times \mathbf{c}_{12}, \mathbf{c}_{21} \times \mathbf{c}_{22})\} = (\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) = \mathbf{S} \end{aligned}$$

Thus, the result holds.

5. If $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}_1$ and $\mathbf{G}' = \forall \mathbf{X}. \mathbf{G}'_1$, then $\varepsilon = \forall \mathbf{X}. \varepsilon'$, where $\varepsilon' = \text{schm}(\varepsilon)$. Let us suppose that $\mathbf{S} = (\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon$. We know that $\mathbf{c}_1 : \mathbf{G}'' \sqsubseteq \forall \mathbf{X}. \mathbf{G}_1$ and $\mathbf{c}_2 : \mathbf{G}'' \sqsubseteq \forall \mathbf{X}. \mathbf{G}'_1$. Therefore, by Lemma B.11, we get that $\mathbf{G}'' = \forall \mathbf{X}. \mathbf{G}'_1$, $\mathbf{c}_1 = \forall \mathbf{X}. \mathbf{c}_{11}$ and $\mathbf{c}_2 = \forall \mathbf{X}. \mathbf{c}_{21}$. Thus, we know that

$$\begin{aligned} & \bullet \{\mathbf{S}'\} = \text{schm}(\mathbf{S}) = \{(\mathbf{G}'_1, \mathbf{c}_{11}, \mathbf{c}_{21})\} \\ & \bullet \forall \mathbf{X}. \{\mathbf{S}'\} = \{(\forall \mathbf{X}. \mathbf{G}'_1, \forall \mathbf{X}. \mathbf{c}_{11}, \forall \mathbf{X}. \mathbf{c}_{21})\} = (\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) = \mathbf{S} \end{aligned}$$

Thus, the result holds.

6. If $\mathbf{G} = \mathbf{B}$ and $\mathbf{G}' = ?_\delta$, then $\forall \mathbf{S} \in \varepsilon$, $\mathbf{S} = (\mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{B}})$, or $\mathbf{S} = (\mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{X}})$ and $\mathbf{X} : \mathbf{B} \in \delta$. Let us suppose that $(\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon$. We know that $\mathbf{c}_1 : \mathbf{G}'' \sqsubseteq \mathbf{B}$ and $\mathbf{c}_2 : \mathbf{G}'' \sqsubseteq ?_\delta$. Therefore, by Lemma B.11, we get that $\mathbf{G}'' = \mathbf{B}$, $\mathbf{c}_1 = \mathbf{B}$, and $\mathbf{c}_2 = \text{inj}_{\mathbf{B}}$ or $\mathbf{c}_2 = \text{inj}_{\mathbf{X}}$ and $\mathbf{X} : \mathbf{B} \in \delta$. Thus, the result holds.
7. If $\mathbf{G} = \mathbf{X}$ and $\mathbf{G}' = ?_\delta$, then $\forall \mathbf{S} \in \varepsilon$, $\mathbf{S} = (\mathbf{X}, \mathbf{X}, \text{inj}_{\mathbf{X}})$ and $\mathbf{X} : \mathbf{X} \in \delta$, or $\mathbf{S} = (\mathbf{X}, \mathbf{X}, \text{inj}_{\mathbf{Y}})$, $\mathbf{X} \neq \mathbf{Y}$, $\mathbf{X} : \mathbf{X} \in \delta$, and $\mathbf{Y} : \mathbf{X} \in \delta$. Let us suppose that $(\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon$. We know that $\mathbf{c}_1 : \mathbf{G}'' \sqsubseteq \mathbf{X}$ and $\mathbf{c}_2 : \mathbf{G}'' \sqsubseteq ?_\delta$. Therefore, by Lemma B.11, we get that $\mathbf{G}'' = \mathbf{X}$, $\mathbf{c}_1 = \mathbf{X}$, and $\mathbf{c}_2 = \text{inj}_{\mathbf{X}}$ and $\mathbf{X} : \mathbf{X} \in \delta$, or $\mathbf{c}_2 = \text{inj}_{\mathbf{Y}}$ and $\mathbf{X} \neq \mathbf{Y}$, $\mathbf{X} : \mathbf{X} \in \delta$, and $\mathbf{Y} : \mathbf{X} \in \delta$. Thus, the result holds.

8. If $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\mathbf{G}' = ?_\delta$, then $\varepsilon = \text{inj}_{\rightarrow 2}(\varepsilon_1) = \varepsilon_1 \circledast \varepsilon_2$, where $\varepsilon_1 = \text{extract}_{\rightarrow 2}(\varepsilon)$ and $\varepsilon_2 = [(?_\delta \rightarrow ?_\delta, \text{inj}_? \rightarrow \text{inj}_?, \text{inj}_{\rightarrow}(\text{inj}_? \rightarrow \text{inj}_?))]$. Let us suppose that $\mathbf{S} = (\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon$. We know that $\mathbf{c}_1 : \mathbf{G}'' \sqsubseteq \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\mathbf{c}_2 : \mathbf{G}'' \sqsubseteq ?_\delta$. Therefore, by Lemma B.11, we get that $\mathbf{G}'' = \mathbf{G}_1'' \rightarrow \mathbf{G}_2''$, $\mathbf{c}_1 = \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}$ and $\mathbf{c}_2 = \text{inj}_{\rightarrow}(\mathbf{c}_{21} \rightarrow \mathbf{c}_{22})$. Thus, we know that

- $\{\mathbf{S}_1\} = \text{extract}_{\rightarrow 2}(\{\mathbf{S}\}) = \{(\mathbf{G}_1'' \rightarrow \mathbf{G}_2'', \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22})\}$
- $\{\mathbf{S}_1\} \circledast \varepsilon_2 = \{\mathbf{S}\}$
- $\text{inj}_{\rightarrow 2}(\{\mathbf{S}_1\}) = \{(\mathbf{G}_1'' \rightarrow \mathbf{G}_2'', \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \text{inj}_{\rightarrow}(\mathbf{c}_{21} \rightarrow \mathbf{c}_{22}))\} = (\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) = \mathbf{S}$

Thus, the result holds.

9. If $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$ and $\mathbf{G}' = ?_\delta$, then $\varepsilon = \text{inj}_{\times 2}(\varepsilon_1) = \varepsilon_1 \circledast \varepsilon_2$, where $\varepsilon_1 = \text{extract}_{\times 2}(\varepsilon)$ and $\varepsilon_2 = [(?_\delta \times ?_\delta, \text{inj}_? \times \text{inj}_?, \text{inj}_{\times}(\text{inj}_? \times \text{inj}_?))]$. Let us suppose that $\mathbf{S} = (\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon$. We know that $\mathbf{c}_1 : \mathbf{G}'' \sqsubseteq \mathbf{G}_1 \times \mathbf{G}_2$ and $\mathbf{c}_2 : \mathbf{G}'' \sqsubseteq ?_\delta$. Therefore, by Lemma B.11, we get that $\mathbf{G}'' = \mathbf{G}_1'' \times \mathbf{G}_2''$, $\mathbf{c}_1 = \mathbf{c}_{11} \times \mathbf{c}_{12}$ and $\mathbf{c}_2 = \text{inj}_{\times}(\mathbf{c}_{21} \times \mathbf{c}_{22})$. Thus, we know that

- $\{\mathbf{S}_1\} = \text{extract}_{\times 2}(\{\mathbf{S}\}) = \{(\mathbf{G}_1'' \times \mathbf{G}_2'', \mathbf{c}_{11} \times \mathbf{c}_{12}, \mathbf{c}_{21} \times \mathbf{c}_{22})\}$
- $\{\mathbf{S}_1\} \circledast \varepsilon_2 = \{\mathbf{S}\}$
- $\text{inj}_{\times 2}(\{\mathbf{S}_1\}) = \{(\mathbf{G}_1'' \times \mathbf{G}_2'', \mathbf{c}_{11} \times \mathbf{c}_{12}, \text{inj}_{\times}(\mathbf{c}_{21} \times \mathbf{c}_{22}))\} = (\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) = \mathbf{S}$

Thus, the result holds.

10. If $\mathbf{G} = \forall \mathbf{X}.\mathbf{G}_1$ and $\mathbf{G}' = ?_\delta$, then $\varepsilon = \text{inj}_{\forall 2}(\varepsilon_1) = \varepsilon_1 \circledast \varepsilon_2$, where $\varepsilon_1 = \text{extract}_{\forall 2}(\varepsilon)$ and $\varepsilon_2 = [(\forall \mathbf{X}.\text{?}_\delta, \mathbf{x}:\mathbf{X}, \forall \mathbf{X}.\text{inj}_?, \text{inj}_{\forall}(\forall \mathbf{X}.\text{inj}_?))]$. Let us suppose that $\mathbf{S} = (\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon$. We know that $\mathbf{c}_1 : \mathbf{G}'' \sqsubseteq \forall \mathbf{X}.\mathbf{G}_1$ and $\mathbf{c}_2 : \mathbf{G}'' \sqsubseteq ?_\delta$. Therefore, by Lemma B.11, we get that $\mathbf{G}'' = \forall \mathbf{X}.\mathbf{G}_1''$, $\mathbf{c}_1 = \forall \mathbf{X}.\mathbf{c}_{11}$ and $\mathbf{c}_2 = \text{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_{21})$. Thus, we know that

- $\{\mathbf{S}_1\} = \text{extract}_{\forall 2}(\{\mathbf{S}\}) = \{(\forall \mathbf{X}.\mathbf{G}_1'', \forall \mathbf{X}.\mathbf{c}_{11}, \forall \mathbf{X}.\mathbf{c}_{21})\}$
- $\{\mathbf{S}_1\} \circledast \varepsilon_2 = \{\mathbf{S}\}$
- $\text{inj}_{\forall 2}(\{\mathbf{S}_1\}) = \{(\forall \mathbf{X}.\mathbf{G}_1'', \forall \mathbf{X}.\mathbf{c}_{11}, \text{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_{21}))\} = (\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) = \mathbf{S}$

Thus, the result holds.

11. If $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \mathbf{B}$, then $\forall \mathbf{S} \in \varepsilon$, $\mathbf{S} = (\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})$, or $\mathbf{S} = (\mathbf{B}, \text{inj}_{\mathbf{X}}, \mathbf{B})$ and $\mathbf{X} : \mathbf{B} \in \delta$. Similar to the inverse case.
12. If $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \mathbf{X}$, then $\forall \mathbf{S} \in \varepsilon$, $\mathbf{S} = (\mathbf{X}, \text{inj}_{\mathbf{X}}, \mathbf{X})$ and $\mathbf{X} : \mathbf{X} \in \delta$, or $\mathbf{S} = (\mathbf{X}, \text{inj}_{\mathbf{Y}}, \mathbf{X})$, $\mathbf{X} \neq \mathbf{Y}$, $\mathbf{X} : \mathbf{X} \in \delta$, and $\mathbf{Y} : \mathbf{X} \in \delta$. Similar to the inverse case.
13. If $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$, then $\varepsilon = \text{inj}_{\rightarrow 1}(\varepsilon_1) = \varepsilon_2 \circledast \varepsilon_1$, where $\varepsilon_1 = \text{extract}_{\rightarrow 1}(\varepsilon)$, $\vdash \varepsilon_2 : \delta \rightarrow ?_\delta \rightarrow ?_\delta$ and $\varepsilon_2 = [(?_\delta \rightarrow ?_\delta, (\text{inj}_? \rightarrow \text{inj}_?)\text{inj}_{\rightarrow}, \text{inj}_? \rightarrow \text{inj}_?)]$. Similar to the inverse case.
14. If $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \mathbf{G}'_1 \times \mathbf{G}'_2$, then $\varepsilon = \text{inj}_{\times 1}(\varepsilon_1) = \varepsilon_2 \circledast \varepsilon_1$, where $\varepsilon_1 = \text{extract}_{\times 1}(\varepsilon)$, $\vdash \varepsilon_2 : \delta \rightarrow ?_\delta \times ?_\delta$ and $\varepsilon_2 = [(?_\delta \times ?_\delta, \text{inj}_{\times}(\text{inj}_? \times \text{inj}_?), \text{inj}_? \times \text{inj}_?)]$. Similar to the inverse case.
15. If $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \forall \mathbf{X}.\mathbf{G}'_1$, then $\varepsilon = \text{inj}_{\forall 1}(\varepsilon_1) = \varepsilon_2 \circledast \varepsilon_1$, where $\varepsilon_1 = \text{extract}_{\forall 1}(\varepsilon)$, $\vdash \varepsilon_2 : \delta \rightarrow \forall \mathbf{X}.\text{?}_\delta, \mathbf{x}:\mathbf{X}$ and $\varepsilon_2 = [(\forall \mathbf{X}.\text{?}_\delta, \mathbf{x}:\mathbf{X}, \text{inj}_{\forall}(\forall \mathbf{X}.\text{inj}_?), \forall \mathbf{X}.\text{inj}_?)]$. Similar to the inverse case.

□

Lemma B.42 *If $\varepsilon : ?_\delta \sim \mathbf{G}$, with $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$, or $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$, or $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}_1$, then $\varepsilon = \varepsilon_1 \circledast \varepsilon_2$, where $\mathcal{S}(?_\delta, \mathbf{G}) = \mathbf{G}_R$, $c(\mathbf{G}) = c$, $\vdash \varepsilon_1 : \delta \rightarrow \mathbf{G}_R$ and $\varepsilon_2 = \text{extract}_{c1}(\varepsilon)$.*

PROOF. Directly by Lemma B.41. □

Lemma B.43 *If $\varepsilon : \mathbf{B} \sim ?_\delta$, $\varepsilon' : ?_\delta \sim \mathbf{B}$ and $\varepsilon \circledast \varepsilon'$ is defined, then $\varepsilon \circledast \varepsilon' = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\}$.*

PROOF. Since $\varepsilon \circledast \varepsilon'$ is defined, then we know that $\varepsilon \circledast \varepsilon' : \mathbf{B} \sim \mathbf{B}$. Then, by Lemma B.41, we know that $\varepsilon \circledast \varepsilon' = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\}$, and the results holds. □

Lemma B.44 *If $\Delta; \Gamma \vdash \varepsilon \mathbf{u} :: ?_\delta : ?_\delta$, $\varepsilon : \mathbf{G}_u \sim ?_\delta$, $\varepsilon' : ?_\delta \sim \mathbf{B}$ and $\varepsilon \circledast \varepsilon'$ is defined, then $\mathbf{G}_u = \mathbf{B}$ and $\mathbf{u} = \mathbf{b}$.*

PROOF. Since $\varepsilon \circledast \varepsilon'$ is defined, then we know that $\varepsilon \circledast \varepsilon' : \mathbf{G}_u \sim \mathbf{B}$. Then, $\mathbf{G}_u = ?_{\delta'}$ or $\mathbf{G}_u = \mathbf{B}$. Since $\Delta; \Gamma \vdash \varepsilon \mathbf{u} :: ?_\delta : ?_\delta$, $\varepsilon : \mathbf{G}_u \sim ?_\delta$, we know by (\mathbf{Gasc}) rule that $\Delta; \Gamma \vdash \mathbf{u} : \mathbf{G}_u$. Therefore, \mathbf{G}_u cannot be $?_{\delta'}$, thus $\mathbf{G}_u = \mathbf{B}$ and $\mathbf{u} = \mathbf{b}$. Finally, the result holds. □

Lemma B.45 *If $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$ and $j < n$, then $\triangleright^j (n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$.*

PROOF. We proceed by induction on n and \mathbf{G} . We know that $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$ and $j < n$. We are required to prove that $\triangleright^j (n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$. Or what is the same, we are required to prove that $(n_j, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$, where $n_j = n - j \leq n$.

Case ($\mathbf{G} = \mathbf{B}$). Immediate from the definition of $\mathcal{V}_\rho[\mathbf{B}]$ and $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{B}]$.

Case ($\mathbf{G} = \mathbf{X}$). Immediate from definitions of $\mathcal{V}_\rho[\mathbf{X}] = \rho.R(\mathbf{X}) \in \text{REL}[\rho_1(\mathbf{X}), \rho_2(\mathbf{X})]$ and $\text{REL}[\rho_1(\mathbf{X}), \rho_2(\mathbf{X})]$, and $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{X}]$.

Case ($\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$). Consider arbitrary n' , \mathbf{v}'_1 and \mathbf{v}'_2 such that $n' \leq n_j$ and $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[\mathbf{G}_1]$. It suffices to show that $(n', \mathbf{v}_1 \mathbf{v}'_1, \mathbf{v}_2 \mathbf{v}'_2) \in \mathcal{T}_\rho[\mathbf{G}_2]$. Instantiate $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{T}_\rho[\mathbf{G}_1 \rightarrow \mathbf{G}_2]$ with n' , \mathbf{v}'_1 and \mathbf{v}'_2 , noting that $n' \leq n_j \leq n$. Hence, we have $(n', \mathbf{v}_1 \mathbf{v}'_1, \mathbf{v}_2 \mathbf{v}'_2) \in \mathcal{T}_\rho[\mathbf{G}_2]$ as we are required to show.

Case ($\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$). The proof of this case is straightforward.

Case ($\mathbf{G} = \forall \mathbf{X}. \mathbf{G}_1$). Consider arbitrary \mathbf{B}_1 , \mathbf{B}_2 and R such that $\mathbf{B}_1 \vdash \mathbf{B}_2$ and $R \in \text{REL}[\mathbf{B}_1, \mathbf{B}_2]$. It suffices to show that $(n_j, \mathbf{v}_1 [\mathbf{B}_1], \mathbf{v}_2 [\mathbf{B}_2]) \in \mathcal{T}_{(\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))}[\mathbf{G}_1]$. Or what is the same, we are required to prove that $\forall i < n_j \leq n$,

$$\begin{aligned} (\forall \mathbf{v}_{1f}. \mathbf{v}_1 [\mathbf{B}_1] \mapsto^i \mathbf{v}_{1f} \Rightarrow \exists \mathbf{v}_{2f}. \mathbf{v}_2 [\mathbf{B}_2] \mapsto^* \mathbf{v}_{2f} \wedge \triangleright^i (n_j, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{(\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))}[\mathbf{G}_1]) \wedge \\ (\mathbf{v}_1 [\mathbf{B}_1] \mapsto^i \mathbf{error} \Rightarrow \mathbf{v}_2 [\mathbf{B}_2] \mapsto^* \mathbf{error}) \end{aligned}$$

Instantiate $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\forall \mathbf{X}. \mathbf{G}_1]$, with $\mathbf{B}_1 \vdash \mathbf{B}_2$ and $R \in \text{REL}[\mathbf{B}_1, \mathbf{B}_2]$. Then, we get that $\forall i < n$,

$$\begin{aligned} (\forall \mathbf{v}_{1f}. \mathbf{v}_1 [\mathbf{B}_1] \mapsto^i \mathbf{v}_{1f} \Rightarrow \exists \mathbf{v}_{2f}. \mathbf{v}_2 [\mathbf{B}_2] \mapsto^* \mathbf{v}_{2f} \wedge \triangleright^i (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{(\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))}[\mathbf{G}_1]) \wedge \\ (\mathbf{v}_1 [\mathbf{B}_1] \mapsto^i \mathbf{error} \Rightarrow \mathbf{v}_2 [\mathbf{B}_2] \mapsto^* \mathbf{error}) \end{aligned}$$

If $\mathbf{v}_1 [\mathbf{B}_1] \mapsto^i \mathbf{error}$ then $\mathbf{v}_2 [\mathbf{B}_2] \mapsto^* \mathbf{error}$ and the result holds. If $\mathbf{v}_1 [\mathbf{B}_1] \mapsto^i \mathbf{v}_{1f}$ then we know that $\exists \mathbf{v}_{2f}. \mathbf{v}_2 [\mathbf{B}_2] \mapsto^* \mathbf{v}_{2f} \wedge \triangleright^i (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{(\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))}[\mathbf{G}_1]$. By the induction hypothesis on \mathbf{G}_1 ($n - i < n$), with $\triangleright^i (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{(\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))}[\mathbf{G}_1]$, we get that $\triangleright^i (n_j, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{(\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))}[\mathbf{G}_1]$ as we are required to show.

Case ($\mathbf{G} = ?_\delta$). It suffices to show that $\forall \mathbf{G}_R, \varepsilon$ such that $\vdash \varepsilon : \delta \rightarrow \mathbf{G}_R, \varepsilon : ?_\delta \sim \mathbf{G}'_R$, we get

$$(n_j, \rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}'_R), \rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}'_R)) \in \mathcal{J}_\rho[\mathbf{G}'_R]$$

Therefore, we are required to prove that $\forall i < n_j \leq n$,

$$\begin{aligned} (\forall \mathbf{v}_{1f}. \rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}'_R) \mapsto^i \mathbf{v}_{1f} \Rightarrow \exists \mathbf{v}_{2f}. \rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}'_R) \mapsto^* \mathbf{v}_{2f} \wedge \triangleright^i (n_j, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\mathbf{G}'_R]) \wedge \\ (\rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}'_R) \mapsto^i \mathbf{error} \Rightarrow \rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}'_R) \mapsto^* \mathbf{error}) \end{aligned}$$

Instantiate $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[?_\delta]$, with \mathbf{G}_R . Then, we get that

$$(n, \rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}'_R), \rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}'_R)) \in \mathcal{J}_\rho[\mathbf{G}'_R]$$

Thus, if $\rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}'_R) \mapsto^i \mathbf{error}$ then we know that $\rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}'_R) \mapsto^* \mathbf{error}$, and the result holds. Otherwise, if $\rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}'_R) \mapsto^i \mathbf{v}_{1f}$, we know that $\rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}'_R) \mapsto^* \mathbf{v}_{2f}$ and $\triangleright^i (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\mathbf{G}'_R]$. Thus, we are required to prove that $\triangleright^i (n_j, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\mathbf{G}'_R]$. We have five cases to consider.

- $\mathbf{G}'_R = \mathbf{B}$. Immediate from the definition of $\mathcal{V}_\rho[\mathbf{B}]$ and $\triangleright^i (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\mathbf{B}]$.
- $\mathbf{G}'_R = \mathbf{X}$. Immediate from definitions of $\mathcal{V}_\rho[\mathbf{X}] = \rho.R(\mathbf{X}) \in \text{REL}[\rho_1(\mathbf{X}), \rho_2(\mathbf{X})]$ and $\text{REL}[\rho_1(\mathbf{X}), \rho_2(\mathbf{X})]$, and $\triangleright^i (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\mathbf{X}]$.
- $\mathbf{G}'_R = ?_\delta \rightarrow ?_\delta$. Consider arbitrary n', \mathbf{v}'_1 and \mathbf{v}'_2 such that $n' \leq n_j - i$ and $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[?_\delta]$. It suffices to show that $(n', \mathbf{v}_{1f} \mathbf{v}'_1, \mathbf{v}_{2f} \mathbf{v}'_2) \in \mathcal{J}_\rho[?_\delta]$. Instantiate $\triangleright^i (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[?_\delta \rightarrow ?_\delta]$ with n', \mathbf{v}'_1 and \mathbf{v}'_2 , noting that $n' \leq n_j - i \leq n - i$. Hence, we have $(n', \mathbf{v}_1 \mathbf{v}'_1, \mathbf{v}_2 \mathbf{v}'_2) \in \mathcal{J}_\rho[?_\delta]$ as we are required to show.
- $\mathbf{G}'_R = ?_\delta \times ?_\delta$. The proof of this case is straightforward.
- $\mathbf{G}'_R = \forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}$. Consider arbitrary $\mathbf{B}_1, \mathbf{B}_2$ and R such that $\mathbf{B}_1, \vdash \mathbf{B}_2$ and $R \in \text{REL}[\mathbf{B}_1, \mathbf{B}_2]$. It suffices to show that $\triangleright^i (n_j, \mathbf{v}_{1f} [\mathbf{B}_1], \mathbf{v}_{2f} [\mathbf{B}_2]) \in \mathcal{J}_{(\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))}[?_\delta, \mathbf{X} : \mathbf{X}]$. Or what is the same, we are required to prove that $\forall i' < n_j - i \leq n - i$,

$$\begin{aligned} (\forall \mathbf{v}'_{1f}. \mathbf{v}_{1f} [\mathbf{B}_1] \mapsto^{i'} \mathbf{v}'_{1f} \Rightarrow \exists \mathbf{v}'_{2f}. \mathbf{v}_{2f} [\mathbf{B}_2] \mapsto^* \mathbf{v}'_{2f} \wedge \triangleright^{i+i'} (n_j, \mathbf{v}'_{1f}, \mathbf{v}'_{2f}) \in \mathcal{V}_{(\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))}[?_\delta, \mathbf{X} : \mathbf{X}]) \wedge \\ (\mathbf{v}_{1f} [\mathbf{B}_1] \mapsto^{i'} \mathbf{error} \Rightarrow \mathbf{v}_{2f} [\mathbf{B}_2] \mapsto^* \mathbf{error}) \end{aligned}$$

Instantiate $\triangleright^i (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}]$, with $\mathbf{B}_1, \vdash \mathbf{B}_2$ and $R \in \text{REL}[\mathbf{B}_1, \mathbf{B}_2]$. Then, we get that $\forall i' < n - i$,

$$\begin{aligned} (\forall \mathbf{v}'_{1f}. \mathbf{v}_{1f} [\mathbf{B}_1] \mapsto^{i'} \mathbf{v}'_{1f} \Rightarrow \exists \mathbf{v}'_{2f}. \mathbf{v}_{2f} [\mathbf{B}_2] \mapsto^* \mathbf{v}'_{2f} \wedge \triangleright^{i+i'} (n, \mathbf{v}'_{1f}, \mathbf{v}'_{2f}) \in \mathcal{V}_{(\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))}[?_\delta, \mathbf{X} : \mathbf{X}]) \wedge \\ (\mathbf{v}_{1f} [\mathbf{B}_1] \mapsto^{i'} \mathbf{error} \Rightarrow \mathbf{v}_{2f} [\mathbf{B}_2] \mapsto^* \mathbf{error}) \end{aligned}$$

If $\mathbf{v}_{1f} [\mathbf{B}_1] \mapsto^{i'} \mathbf{error}$ then $\mathbf{v}_{2f} [\mathbf{B}_2] \mapsto^* \mathbf{error}$ and the result holds. If $\mathbf{v}_{1f} [\mathbf{B}_1] \mapsto^{i'} \mathbf{v}'_{1f}$ then we know that $\exists \mathbf{v}'_{2f}. \mathbf{v}_{2f} [\mathbf{B}_2] \mapsto^* \mathbf{v}'_{2f} \wedge \triangleright^{i+i'} (n, \mathbf{v}'_{1f}, \mathbf{v}'_{2f}) \in \mathcal{V}_{(\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))}[?_\delta, \mathbf{X} : \mathbf{X}]$. By the induction hypothesis on $n - i - i' < n$, with $\triangleright^{i+i'} (n, \mathbf{v}'_{1f}, \mathbf{v}'_{2f}) \in \mathcal{V}_{(\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))}[?_\delta, \mathbf{X} : \mathbf{X}]$, we get that $\triangleright^{i+i'} (n_j, \mathbf{v}'_{1f}, \mathbf{v}'_{2f}) \in \mathcal{V}_{(\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))}[?_\delta, \mathbf{X} : \mathbf{X}]$ as we are required to show.

□

Lemma B.46 Let $\Delta \vdash \mathbf{G}$, $\rho' = (\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))$ and $(n, \rho') \in \mathcal{D}[\Delta, \mathbf{X}]$. Then

1. $\mathcal{V}_\rho[\mathbf{G}] = \mathcal{V}_{\rho'}[\mathbf{G}]$.
2. $\mathcal{J}_\rho[\mathbf{G}] = \mathcal{J}_{\rho'}[\mathbf{G}]$.

PROOF. The proof proceeds first proving (1) and then (2), by induction on n and \mathbf{G} .

1. We consider the cases for \mathbf{G} . In each case, we may equivalently show that $\mathcal{V}_\rho[\mathbf{G}] = \mathcal{V}_{\rho'}[\mathbf{G}]$. Or what is the same

$$(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}] \iff (n, \mathbf{v}_1, \mathbf{v}_2) \mathcal{V}_{\rho'}[\mathbf{G}]$$

Case ($\mathbf{G} = \mathbf{B}$). Immediate from the definition of $\mathcal{V}_\rho[\mathbf{B}]$, Lemma B.72 and $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{B}]$.

Case ($\mathbf{G} = \mathbf{Y}$). Note that $\mathbf{Y} \neq \mathbf{X}$. Immediate from definitions of $\mathcal{V}_\rho[\mathbf{Y}] = \rho.R(\mathbf{Y}) = \rho'.R(\mathbf{Y}) = \mathcal{V}_{\rho'}[\mathbf{Y}]$.

Case ($\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$). We first prove the \implies direction. Consider arbitrary n' , \mathbf{v}'_1 and \mathbf{v}'_2 such that $n' \leq n$ and $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_{\rho'}[\mathbf{G}_1]$. It suffices to show that $(n', \mathbf{v}_1 \mathbf{v}'_1, \mathbf{v}_2 \mathbf{v}'_2) \in \mathcal{J}_{\rho'}[\mathbf{G}_2]$. Instantiate $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}_1 \rightarrow \mathbf{G}_2]$ with n' , \mathbf{v}'_1 and \mathbf{v}'_2 , noting that $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[\mathbf{G}_1]$ by the inductive hypothesis of part (1) for \mathbf{G}_1 . Hence, we have $(n', \mathbf{v}_1 \mathbf{v}'_1, \mathbf{v}_2 \mathbf{v}'_2) \in \mathcal{J}_\rho[\mathbf{G}_2]$. By the inductive hypothesis of part (2) for \mathbf{G}_2 , we have that $(n', \mathbf{v}_1 \mathbf{v}'_1, \mathbf{v}_2 \mathbf{v}'_2) \in \mathcal{J}_{\rho'}[\mathbf{G}_2]$, as we were required to show. The \impliedby direction is analogous.

Case ($\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$). The proof of this case is straightforward.

Case ($\mathbf{G} = \forall \mathbf{Y}. \mathbf{G}_1$). We first prove the \implies direction. Consider arbitrary \mathbf{B}'_1 , \mathbf{B}'_2 and R' such that $\mathbf{B}'_1 \vdash \mathbf{B}'_2$ and $R' \in \text{REL}[\mathbf{B}'_1, \mathbf{B}'_2]$. It suffices to show that $(n, \mathbf{v}_1 [\mathbf{B}'_1], \mathbf{v}_2 [\mathbf{B}'_2]) \in \mathcal{J}_{\rho'; \mathbf{Y} \mapsto (\mathbf{B}'_1, \mathbf{B}'_2, R')}[\mathbf{G}_1]$. Instantiate $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\forall \mathbf{Y}. \mathbf{G}_1]$, with $\mathbf{B}'_1 \vdash \mathbf{B}'_2$ and $R' \in \text{REL}[\mathbf{B}'_1, \mathbf{B}'_2]$. Then, we get that $(n, \mathbf{v}_1 [\mathbf{B}'_1], \mathbf{v}_2 [\mathbf{B}'_2]) \in \mathcal{J}_{\rho; \mathbf{Y} \mapsto (\mathbf{B}'_1, \mathbf{B}'_2, R')}[\mathbf{G}_1]$. By the inductive hypothesis of part (2) for \mathbf{G}_1 , we have that $(n, \mathbf{v}_1 [\mathbf{B}'_1], \mathbf{v}'_2 [\mathbf{B}'_2]) \in \mathcal{J}_{\rho'; \mathbf{Y} \mapsto (\mathbf{B}'_1, \mathbf{B}'_2, R')}[\mathbf{G}_1]$, as we were required to show. The \impliedby direction is analogous.

Case ($\mathbf{G} = ?_\delta$). We first prove the \implies direction. It suffices to show that $\forall \mathbf{G}_R$ and $\vdash \varepsilon : \delta \rightarrow \mathbf{G}_R$, we get

$$(n, \rho'_1(\varepsilon) \mathbf{v}_1 :: \rho'_1(\mathbf{G}_R), \rho'_2(\varepsilon) \mathbf{v}_2 :: \rho'_2(\mathbf{G}_R)) \in \mathcal{J}_{\rho'}[\mathbf{G}_R]$$

or what is the same

$$(n, \rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}_R), \rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}_R)) \in \mathcal{J}_{\rho'}[\mathbf{G}_R]$$

Therefore, we are required to prove that

$$(\forall \mathbf{v}_{1f}. \rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}_R) \mapsto^0 \mathbf{v}_{1f} \Rightarrow \exists \mathbf{v}_{2f}. \rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}_R) \mapsto^* \mathbf{v}_{2f} \wedge (n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho'}[\mathbf{G}_R]) \wedge$$

$$(\rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}_R) \mapsto^0 \mathbf{error} \Rightarrow \rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}_R) \mapsto^* \mathbf{error})$$

Instantiate $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[?_\delta]$, with \mathbf{G}_R . Then, we get that

$$(n, \rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}_R), \rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}_R)) \in \mathcal{J}_\rho[\mathbf{G}_R]$$

Thus, if $\rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}_R) \mapsto^0 \mathbf{error}$ then we know that $\rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}_R) \mapsto^* \mathbf{error}$, and the result holds. Otherwise, if $\rho_1(\varepsilon) \mathbf{v}_1 :: \rho_1(\mathbf{G}_R) \mapsto^0 \mathbf{v}_{1f}$, we know that $\rho_2(\varepsilon) \mathbf{v}_2 :: \rho_2(\mathbf{G}_R) \mapsto^* \mathbf{v}_{2f}$ and $(n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho'}[\mathbf{G}_R]$. Thus, we are required to prove that $(n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_{\rho'}[\mathbf{G}_R]$. We have five cases to consider.

- $\mathbf{G}_R = \mathbf{B}$. Immediate from the definition of $\mathcal{V}_\rho[\mathbf{B}]$, Lemma B.72 and $(n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\mathbf{B}]$.
- $\mathbf{G}_R = \mathbf{Y}$. Note that $\mathbf{Y} \neq \mathbf{X}$. Immediate from definitions of $\mathcal{V}_\rho[\mathbf{Y}] = \rho.R(\mathbf{Y}) = \rho'.R(\mathbf{Y}) = \mathcal{V}_{\rho'}[\mathbf{Y}]$.
- $\mathbf{G}_R = ?_\delta \rightarrow ?_\delta$. Consider arbitrary n' , \mathbf{v}'_1 and \mathbf{v}'_2 such that $n' \leq n$ and $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[?_\delta]$. It suffices to show that $(n', \mathbf{v}_{1f}, \mathbf{v}_{2f}, \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{J}_{\rho'}[?_\delta]$. Or what is the same that $\forall i < n'$,

$$(\forall \mathbf{v}'_{1f}, \mathbf{v}_{1f} \mathbf{v}'_1 \mapsto^i \mathbf{v}'_{1f} \Rightarrow \exists \mathbf{v}'_{2f}, \mathbf{v}_{2f} \mathbf{v}'_2 \mapsto^* \mathbf{v}'_{2f} \wedge \triangleright^i (n', \mathbf{v}'_{1f}, \mathbf{v}'_{2f}) \in \mathcal{V}_{\rho'}[?_\delta]) \wedge$$

$$(\mathbf{v}_{1f} \mathbf{v}'_1 \mapsto^i \mathbf{error} \Rightarrow \mathbf{v}_{2f} \mathbf{v}'_2 \mapsto^* \mathbf{error})$$

Instantiate $(n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[?_\delta \rightarrow ?_\delta]$ with n' , \mathbf{v}'_1 and \mathbf{v}'_2 , noting that $\triangleright (n', \mathbf{v}'_1, \mathbf{v}'_2) \in \mathcal{V}_\rho[?_\delta]$, by the inductive hypothesis of part (1) for $n' - 1 < n$. Hence, we have $(n', \mathbf{v}_{1f}, \mathbf{v}'_1, \mathbf{v}_{2f}, \mathbf{v}'_2) \in \mathcal{J}_{\rho'}[?_\delta]$. Thus, both terms fail and the result holds, or we instantiate with i and \mathbf{v}'_{1f} , such that $\mathbf{v}_{1f} \mathbf{v}'_1 \mapsto^i \mathbf{v}'_{1f}$, and therefore there exists \mathbf{v}'_{2f} such that $\mathbf{v}_{2f} \mathbf{v}'_2 \mapsto^* \mathbf{v}'_{2f}$ and $\triangleright^i (n', \mathbf{v}'_{1f}, \mathbf{v}'_{2f}) \in \mathcal{V}_\rho[?_\delta]$. Since $i > 0$, we know that $n' - i < n$. Therefore, by the inductive hypothesis of part (1) for $n' - i < n$, we get that $\triangleright^i (n', \mathbf{v}'_{1f}, \mathbf{v}'_{2f}) \in \mathcal{V}_{\rho'}[?_\delta]$, as we are required to show.

- $\mathbf{G}_R = ?_\delta \times ?_\delta$. The proof of this case is straightforward.
- $\mathbf{G}_R = \forall \mathbf{Y}. ?_\delta, \mathbf{Y} : \mathbf{Y}$. Consider arbitrary \mathbf{B}'_1 , \mathbf{B}'_2 and R such that $\mathbf{B}'_1, \vdash \mathbf{B}'_2$ and $R' \in \text{REL}[\mathbf{B}'_1, \mathbf{B}'_2]$. It suffices to show that $(n, \mathbf{v}_{1f} [\mathbf{B}'_1], \mathbf{v}_{2f} [\mathbf{B}'_2]) \in \mathcal{J}_{(\rho'; \mathbf{Y} \mapsto (\mathbf{B}'_1, \mathbf{B}'_2, R'))}[?_\delta, \mathbf{Y} : \mathbf{Y}]$. Or what is the same, we are required to prove that $\forall i < n$,

$$(\forall \mathbf{v}'_{1f}, \mathbf{v}_{1f} [\mathbf{B}'_1] \mapsto^i \mathbf{v}'_{1f} \Rightarrow \exists \mathbf{v}'_{2f}, \mathbf{v}_{2f} [\mathbf{B}'_2] \mapsto^* \mathbf{v}'_{2f} \wedge \triangleright^i (n, \mathbf{v}'_{1f}, \mathbf{v}'_{2f}) \in \mathcal{V}_{(\rho'; \mathbf{Y} \mapsto (\mathbf{B}'_1, \mathbf{B}'_2, R'))}[?_\delta, \mathbf{Y} : \mathbf{Y}]) \wedge$$

$$(\mathbf{v}_{1f} [\mathbf{B}'_1] \mapsto^i \mathbf{error} \Rightarrow \mathbf{v}_{2f} [\mathbf{B}'_2] \mapsto^* \mathbf{error})$$

Instantiate $(n, \mathbf{v}_{1f}, \mathbf{v}_{2f}) \in \mathcal{V}_\rho[\forall \mathbf{Y}. ?_\delta, \mathbf{Y} : \mathbf{Y}]$, with $\mathbf{B}'_1, \vdash \mathbf{B}'_2$ and $R' \in \text{REL}[\mathbf{B}'_1, \mathbf{B}'_2]$. Then, we get that $\forall i < n$,

$$(\forall \mathbf{v}'_{1f}, \mathbf{v}_{1f} [\mathbf{B}'_1] \mapsto^i \mathbf{v}'_{1f} \Rightarrow \exists \mathbf{v}'_{2f}, \mathbf{v}_{2f} [\mathbf{B}'_2] \mapsto^* \mathbf{v}'_{2f} \wedge \triangleright^i (n, \mathbf{v}'_{1f}, \mathbf{v}'_{2f}) \in \mathcal{V}_{(\rho'; \mathbf{Y} \mapsto (\mathbf{B}'_1, \mathbf{B}'_2, R'))}[?_\delta, \mathbf{Y} : \mathbf{Y}]) \wedge$$

$$(\mathbf{v}_{1f} [\mathbf{B}'_1] \mapsto^i \mathbf{error} \Rightarrow \mathbf{v}_{2f} [\mathbf{B}'_2] \mapsto^* \mathbf{error})$$

If $\mathbf{v}_{1f} [\mathbf{B}'_1] \mapsto^i \mathbf{error}$ then $\mathbf{v}_{2f} [\mathbf{B}'_2] \mapsto^* \mathbf{error}$ and the result holds. If $\mathbf{v}_{1f} [\mathbf{B}'_1] \mapsto^i \mathbf{v}'_{1f}$ then we know that $\exists \mathbf{v}'_{2f}, \mathbf{v}_{2f} [\mathbf{B}'_2] \mapsto^* \mathbf{v}'_{2f} \wedge \triangleright^i (n, \mathbf{v}'_{1f}, \mathbf{v}'_{2f}) \in \mathcal{V}_{(\rho'; \mathbf{Y} \mapsto (\mathbf{B}'_1, \mathbf{B}'_2, R'))}[?_\delta, \mathbf{Y} : \mathbf{Y}]$. Since $i > 0$, we know that $n - i < n$. Therefore, by the inductive hypothesis of part (1) for $n' - i < n$, we get that $\triangleright^i (n, \mathbf{v}'_{1f}, \mathbf{v}'_{2f}) \in \mathcal{V}_{(\rho'; \mathbf{Y} \mapsto (\mathbf{B}'_1, \mathbf{B}'_2, R'))}[?_\delta, \mathbf{Y} : \mathbf{Y}]$ as we are required to show.

The \Leftarrow direction is analogous.

2. We may equivalently show that $\mathcal{J}_\rho[\mathbf{G}] = \mathcal{J}_{\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R)}[\mathbf{G}]$. Or what is the same

$$(n, \mathbf{t}_1, \mathbf{t}_2) \in \mathcal{J}_\rho[\mathbf{G}] \iff (n, \mathbf{t}_1, \mathbf{t}_2) \mathcal{J}_{\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R)}[\mathbf{G}]$$

We first prove the \Leftarrow direction. Assume that $\mathcal{J}_{\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R)}[\mathbf{G}]$. We are required to show that $(n, \mathbf{t}_1, \mathbf{t}_2) \in \mathcal{J}_\rho[\mathbf{G}]$. We proceed by cases on termination of \mathbf{t}_1 .

Case. $\mathbf{t}_1 \mapsto^k \mathbf{v}_{f1}$, where $k < n$. Instantiate the assumption with k and \mathbf{v}_{f1} . We have that there exist some \mathbf{v}_{f2} such that $\mathbf{t}_2 \mapsto^* \mathbf{v}_{f2}$ and $\triangleright^k (n, \mathbf{v}_{f1}, \mathbf{v}_{f2}) \mathcal{V}_{\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R)}[\mathbf{G}]$. Choose \mathbf{v}_{f2} . We have that $\mathbf{t}_2 \mapsto^* \mathbf{v}_{f2}$ and $\triangleright^k (n, \mathbf{v}_{f1}, \mathbf{v}_{f2}) \in \mathcal{J}_\rho[\mathbf{G}]$, by part (1). Therefore, we have that $(n, \mathbf{t}_1, \mathbf{t}_2) \in \mathcal{J}_\rho[\mathbf{G}]$ as we were required to show.

Case. $\mathbf{t}_1 \mapsto^k \mathbf{error}$, where $k < n$. Instantiate the assumption with k . We have that $\mathbf{t}_2 \mapsto^* \mathbf{error}$. Therefore, we also have that $(n, \mathbf{t}_1, \mathbf{t}_2) \in \mathcal{I}_\rho[\mathbf{G}]$, as we were required to show.

The proof for the the \implies direction is identical.

□

Lemma B.47 *If $\varepsilon_1 : \mathbf{G}_1 \rightarrow \mathbf{G}'_1 \sim \mathbf{G} \rightarrow \mathbf{G}'$ and $\varepsilon_2 : \mathbf{G} \rightarrow \mathbf{G}' \sim \mathbf{G}_2 \rightarrow \mathbf{G}'_2$, then $\text{dom}(\varepsilon_1 \circledast \varepsilon_2) = \text{dom}(\varepsilon_2) \circledast \text{dom}(\varepsilon_1)$ and $\text{cod}(\varepsilon_1 \circledast \varepsilon_2) = \text{cod}(\varepsilon_1) \circledast \text{cod}(\varepsilon_2)$.*

Lemma B.48 *If $\varepsilon_1 : \mathbf{G}_1 \times \mathbf{G}'_1 \sim \mathbf{G} \times \mathbf{G}'$ and $\varepsilon_2 : \mathbf{G} \times \mathbf{G}' \sim \mathbf{G}_2 \times \mathbf{G}'_2$, then $\pi_i(\varepsilon_1 \circledast \varepsilon_2) = \pi_i(\varepsilon_1) \circledast \pi_i(\varepsilon_2)$.*

PROOF. We know that $\varepsilon_1 : \mathbf{G}_1 \rightarrow \mathbf{G}'_1 \sim \mathbf{G} \rightarrow \mathbf{G}'$ and $\varepsilon_2 : \mathbf{G} \rightarrow \mathbf{G}' \sim \mathbf{G}_2 \rightarrow \mathbf{G}'_2$. We are required to prove that $\text{dom}(\varepsilon_1 \circledast \varepsilon_2) = \text{dom}(\varepsilon_2) \circledast \text{dom}(\varepsilon_1)$ and $\text{cod}(\varepsilon_1 \circledast \varepsilon_2) = \text{cod}(\varepsilon_1) \circledast \text{cod}(\varepsilon_2)$. We proceed by proving that for all $\mathbf{S}_1 \in \varepsilon_1$ and $\mathbf{S}_2 \in \varepsilon_2$, we get that $\text{dom}(\{\mathbf{S}_1\} \circledast \{\mathbf{S}_2\}) = \text{dom}(\{\mathbf{S}_2\}) \circledast \text{dom}(\{\mathbf{S}_1\})$ and $\text{cod}(\{\mathbf{S}_1\} \circledast \{\mathbf{S}_2\}) = \text{cod}(\{\mathbf{S}_1\}) \circledast \text{cod}(\{\mathbf{S}_2\})$. Since $\varepsilon_1 : \mathbf{G}_1 \rightarrow \mathbf{G}'_1 \sim \mathbf{G} \rightarrow \mathbf{G}'$, $\varepsilon_2 : \mathbf{G} \rightarrow \mathbf{G}' \sim \mathbf{G}_2 \rightarrow \mathbf{G}'_2$ and $\varepsilon_1 \circledast \varepsilon_2 : \mathbf{G}_1 \rightarrow \mathbf{G}'_1 \sim \mathbf{G}_2 \rightarrow \mathbf{G}'_2$, we know that $\{\mathbf{S}_1\} : \mathbf{G}_1 \rightarrow \mathbf{G}'_1 \sim \mathbf{G} \rightarrow \mathbf{G}'$, $\{\mathbf{S}_2\} : \mathbf{G} \rightarrow \mathbf{G}' \sim \mathbf{G}_2 \rightarrow \mathbf{G}'_2$ and $\{\mathbf{S}_1\} \circledast \{\mathbf{S}_2\} : \mathbf{G}_1 \rightarrow \mathbf{G}'_1 \sim \mathbf{G}_2 \rightarrow \mathbf{G}'_2$. By Lemma B.41, we know that $\{\mathbf{S}_1\} = \text{dom}^{-1}(\{\mathbf{S}_1\}) \rightarrow \text{cod}(\{\mathbf{S}_1\})$, $\{\mathbf{S}_2\} = \text{dom}^{-1}(\{\mathbf{S}_2\}) \rightarrow \text{cod}(\{\mathbf{S}_2\})$ and $(\{\mathbf{S}_1\} \circledast \{\mathbf{S}_2\}) = \text{dom}^{-1}(\{\mathbf{S}_1\} \circledast \{\mathbf{S}_2\}) \rightarrow \text{cod}(\{\mathbf{S}_1\} \circledast \{\mathbf{S}_2\})$. Let us define

- $\text{dom}^{-1}(\{\mathbf{S}_1\}) = \{(\mathbf{G}_{11}, \mathbf{c}_{11}, \mathbf{c}'_{11})\}$
- $\text{dom}^{-1}(\{\mathbf{S}_2\}) = \{(\mathbf{G}_{21}, \mathbf{c}'_{21}, \mathbf{c}_{21})\}$
- $\text{cod}(\{\mathbf{S}_1\}) = \{(\mathbf{G}_{12}, \mathbf{c}_{12}, \mathbf{c}'_{12})\}$
- $\text{cod}(\{\mathbf{S}_2\}) = \{(\mathbf{G}_{22}, \mathbf{c}'_{22}, \mathbf{c}_{22})\}$
- $\text{dom}(\{\mathbf{S}_1\}) = \{(\mathbf{G}_{11}, \mathbf{c}'_{11}, \mathbf{c}_{11})\}$
- $\text{dom}(\{\mathbf{S}_2\}) = \{(\mathbf{G}_{21}, \mathbf{c}_{21}, \mathbf{c}'_{21})\}$

Therefore, we know that $\mathbf{S}_1 = (\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \mathbf{c}'_{11} \rightarrow \mathbf{c}'_{12})$ and $\mathbf{S}_2 = (\mathbf{G}_{21} \rightarrow \mathbf{G}_{22}, \mathbf{c}'_{21} \rightarrow \mathbf{c}'_{22}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22})$. By definition of *meet*, we get that

$$\begin{aligned} \text{pullback}(\mathbf{G} \rightarrow \mathbf{G}', (\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}, \mathbf{c}'_{11} \rightarrow \mathbf{c}'_{12}), (\mathbf{G}_{21} \rightarrow \mathbf{G}_{22}, \mathbf{c}'_{21} \rightarrow \mathbf{c}'_{22})) = \\ (\mathbf{G}_3 \rightarrow \mathbf{G}'_3, \mathbf{c}_3 \rightarrow \mathbf{c}'_3, \mathbf{c}_4 \rightarrow \mathbf{c}'_4, \mathbf{c}_5 \rightarrow \mathbf{c}'_5) \end{aligned}$$

where $\text{pullback}(\mathbf{G}, (\mathbf{G}_{11}, \mathbf{c}'_{11}), (\mathbf{G}_{21}, \mathbf{c}'_{21})) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$ and $\text{pullback}(\mathbf{G}', (\mathbf{G}_{12}, \mathbf{c}'_{12}), (\mathbf{G}_{22}, \mathbf{c}'_{22})) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. Therefore, we know that $\{\mathbf{S}_1\} \circledast \{\mathbf{S}_2\} = (\mathbf{G}_3 \rightarrow \mathbf{G}'_3, \mathbf{c}_1 \rightarrow \mathbf{c}_1, \mathbf{c}_2 \rightarrow \mathbf{c}'_2)$, where

- $(\mathbf{c}_3 \rightarrow \mathbf{c}'_3); (\mathbf{c}_{11} \rightarrow \mathbf{c}_{12}) = (\mathbf{c}_1 \rightarrow \mathbf{c}_1)$
- $(\mathbf{c}_4 \rightarrow \mathbf{c}'_4); (\mathbf{c}_{21} \rightarrow \mathbf{c}_{22}) = (\mathbf{c}_2 \rightarrow \mathbf{c}'_2)$
- $\mathbf{c}_3; \mathbf{c}_{11} = \mathbf{c}_1$

- $\mathbf{c}_4; \mathbf{c}_{21} = \mathbf{c}_2$
- $\mathbf{c}'_3; \mathbf{c}_{12} = \mathbf{c}_1$
- $\mathbf{c}'_4; \mathbf{c}_{22} = \mathbf{c}'_2$
- $\text{dom}(\{\mathbf{S}_1\} \circ \{\mathbf{S}_2\}) = \{(\mathbf{G}_3, \mathbf{c}_2, \mathbf{c}_1)\}$
- $\text{cod}(\{\mathbf{S}_1\} \circ \{\mathbf{S}_2\}) = \{(\mathbf{G}'_3, \mathbf{c}_1, \mathbf{c}'_2)\}$

Since $\text{pullback}(\mathbf{G}, (\mathbf{G}_{11}, \mathbf{c}'_{11}), (\mathbf{G}_{21}, \mathbf{c}'_{21})) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$, by Lemma B.67, we get that $\text{pullback}(\mathbf{G}, (\mathbf{G}_{21}, \mathbf{c}'_{21}), (\mathbf{G}_{11}, \mathbf{c}'_{11})) = (\mathbf{G}_3, \mathbf{c}_4, \mathbf{c}_3, \mathbf{c}_5)$. Thus, we get that

- $\text{dom}(\{\mathbf{S}_2\}) \circ \text{dom}(\{\mathbf{S}_1\}) = \{(\mathbf{G}_{21}, \mathbf{c}_{21}, \mathbf{c}'_{21})\} \circ \{(\mathbf{G}_{11}, \mathbf{c}'_{11}, \mathbf{c}_{11})\} = \{(\mathbf{G}_3, \mathbf{c}_2, \mathbf{c}_1)\}$. Note that $\mathbf{c}_4; \mathbf{c}_{21} = \mathbf{c}_2$ and $\mathbf{c}_3; \mathbf{c}_{11} = \mathbf{c}_1$.
- $\text{cod}(\{\mathbf{S}_1\}) \circ \text{cod}(\{\mathbf{S}_2\}) = \text{cod}(\{(\mathbf{G}_{12}, \mathbf{c}_{12}, \mathbf{c}'_{12})\}) \circ \text{cod}(\{(\mathbf{G}_{22}, \mathbf{c}'_{22}, \mathbf{c}_{22})\}) = \{(\mathbf{G}'_3, \mathbf{c}_1, \mathbf{c}'_2)\}$

We are required to prove that

- $\text{dom}(\{\mathbf{S}_1\} \circ \{\mathbf{S}_2\}) = \{(\mathbf{G}_3, \mathbf{c}_2, \mathbf{c}_1)\} = \text{dom}(\{\mathbf{S}_2\}) \circ \text{dom}(\{\mathbf{S}_1\})$
- $\text{cod}(\{\mathbf{S}_1\} \circ \{\mathbf{S}_2\}) = \{(\mathbf{G}'_3, \mathbf{c}_1, \mathbf{c}'_2)\} = \text{cod}(\{\mathbf{S}_1\}) \circ \text{cod}(\{\mathbf{S}_2\})$

Therefore, the result follows immediately. \square

Lemma B.49 *If $\varepsilon_1 : \forall \mathbf{X}. \mathbf{G}_1 \sim \forall \mathbf{X}. \mathbf{G}$ and $\varepsilon_2 : \forall \mathbf{X}. \mathbf{G} \sim \forall \mathbf{X}. \mathbf{G}_2$, then $\text{schm}(\varepsilon_1 \circ \varepsilon_2) = \text{schm}(\varepsilon_1) \circ \text{schm}(\varepsilon_2)$.*

PROOF. We know that $\varepsilon_1 : \forall \mathbf{X}. \mathbf{G}_1 \sim \forall \mathbf{X}. \mathbf{G}$ and $\varepsilon_2 : \forall \mathbf{X}. \mathbf{G} \sim \forall \mathbf{X}. \mathbf{G}_2$. We are required to prove that $\text{schm}(\varepsilon_1 \circ \varepsilon_2) = \text{schm}(\varepsilon_1) \circ \text{schm}(\varepsilon_2)$. We proceed by proving that for all $\mathbf{S}_1 \in \varepsilon_1$ and $\mathbf{S}_2 \in \varepsilon_2$, we get that $\text{schm}(\{\mathbf{S}_1\} \circ \{\mathbf{S}_2\}) = \text{schm}(\{\mathbf{S}_1\}) \circ \text{schm}(\{\mathbf{S}_2\})$. Since $\varepsilon_1 : \forall \mathbf{X}. \mathbf{G}_1 \sim \forall \mathbf{X}. \mathbf{G}$ and $\varepsilon_2 : \forall \mathbf{X}. \mathbf{G} \sim \forall \mathbf{X}. \mathbf{G}_2$ and $\varepsilon_1 \circ \varepsilon_2 : \forall \mathbf{X}. \mathbf{G}_1 \sim \forall \mathbf{X}. \mathbf{G}_2$, we know that $\{\mathbf{S}_1\} : \forall \mathbf{X}. \mathbf{G}_1 \sim \forall \mathbf{X}. \mathbf{G}$, $\{\mathbf{S}_2\} : \forall \mathbf{X}. \mathbf{G} \sim \forall \mathbf{X}. \mathbf{G}_2$ and $\{\mathbf{S}_1\} \circ \{\mathbf{S}_2\} : \forall \mathbf{X}. \mathbf{G}_1 \sim \forall \mathbf{X}. \mathbf{G}_2$. By Lemma B.41, we know that $\{\mathbf{S}_1\} = \forall \mathbf{X}. \text{schm}(\{\mathbf{S}_1\})$, $\{\mathbf{S}_2\} = \forall \mathbf{X}. \text{schm}(\{\mathbf{S}_2\})$ and $(\{\mathbf{S}_1\} \circ \{\mathbf{S}_2\}) = \forall \mathbf{X}. \text{schm}(\{\mathbf{S}_1\} \circ \{\mathbf{S}_2\})$. Let us define

- $\text{schm}(\{\mathbf{S}_1\}) = \{(\mathbf{G}_{12}, \mathbf{c}_{12}, \mathbf{c}'_{12})\}$
- $\text{schm}(\{\mathbf{S}_2\}) = \{(\mathbf{G}_{22}, \mathbf{c}'_{22}, \mathbf{c}_{22})\}$

Therefore, we know that $\mathbf{S}_1 = (\forall \mathbf{X}. \mathbf{G}_{12}, \forall \mathbf{X}. \mathbf{c}_{12}, \forall \mathbf{X}. \mathbf{c}'_{12})$ and $\mathbf{S}_2 = (\forall \mathbf{X}. \mathbf{G}_{22}, \forall \mathbf{X}. \mathbf{c}'_{22}, \forall \mathbf{X}. \mathbf{c}_{22})$. By definition of *meet*, we get that

$$\begin{aligned} \text{pullback}(\forall \mathbf{X}. \mathbf{G}, (\forall \mathbf{X}. \mathbf{G}_{12}, \forall \mathbf{X}. \mathbf{c}'_{12}), (\forall \mathbf{X}. \mathbf{G}_{22}, \forall \mathbf{X}. \mathbf{c}'_{22})) = \\ (\forall \mathbf{X}. \mathbf{G}_3, \forall \mathbf{X}. \mathbf{c}_3, \forall \mathbf{X}. \mathbf{c}_4, \forall \mathbf{X}. \mathbf{c}_5) \end{aligned}$$

where $\text{pullback}(\mathbf{G}', (\mathbf{G}_{12}, \mathbf{c}'_{12}), (\mathbf{G}_{22}, \mathbf{c}'_{22})) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$. Therefore, we know that $\{\mathbf{S}_1\} \circ \{\mathbf{S}_2\} = (\forall \mathbf{X}. \mathbf{G}_3, \forall \mathbf{X}. \mathbf{c}_1, \forall \mathbf{X}. \mathbf{c}_2)$, where

- $(\forall X.c_3);(\forall X.c_{12}) = (\forall X.c_1)$
- $(\forall X.c_4);(\forall X.c_{22}) = (\forall X.c_2)$
- $c_3;c_{12} = c_1$
- $c_4;c_{22} = c_2$
- $\text{schm}(\{S_1\} \circ \{S_2\}) = \{(G_3, c_1, c_2)\}$

Thus, we get that

- $\text{schm}(\{S_1\}) \circ \text{schm}(\{S_2\}) = \text{schm}(\{(\forall X.G_{12}, \forall X.c_{12}, \forall X.c'_{12})\}) \circ \text{schm}(\{(\forall X.G_{22}, \forall X.c'_{22}, \forall X.c_{22})\}) = \{(G_{12}, c_{12}, c'_{12})\} \circ \{(G_{22}, c'_{22}, c_{22})\} \circ \{(G_3, c_1, c_2)\}$

We are required to prove that

- $\text{schm}(\{S_1\} \circ \{S_2\}) = \{(G_3, c_1, c_2)\} = \text{schm}(\{S_1\}) \circ \text{schm}(\{S_2\})$

Therefore, the result follows immediately. \square

Lemma B.50 $\{S_1\} : G_1 \sim G$, $\{S_2\} : G \sim G_2$ and $\Delta \vdash F$, then $(\{S_1\} \circ \{S_2\})[X \mapsto F] = \{S_1\}[X \mapsto F] \circ \{S_2\}[X \mapsto F]$.

PROOF. Let us suppose that $S_1 = (G'_1, c_1, c_1)$ and $S_2 = (G'_2, c'_2, c_2)$. By Lemma B.68, we know that

$$\begin{aligned} \text{pullback}(G, (G'_1, c_1), (G'_2, c'_2)) &= (G_3, c_3, c_4, c_5) \iff \\ \text{pullback}(G[X \mapsto F], (G'_1[X \mapsto F], c_1[X \mapsto F]), (G'_2[X \mapsto F], c'_2[X \mapsto F])) &= \\ (G_3[X \mapsto F], c_3[X \mapsto F], c_4[X \mapsto F], c_5[X \mapsto F]) & \end{aligned}$$

Thus, both combination fail and the result holds, or both combination are defined. Let us suppose that

- $\text{pullback}(G, (G'_1, c_1), (G'_2, c'_2)) = (G_3, c_3, c_4, c_5)$
- $c_3;c_1 = c'_3$
- $c_4;c_2 = c'_4$

Thus, we know that

- $\{S_1\} \circ \{S_2\} = \{(G_3, c'_3, c'_4)\}$
- $(\{S_1\} \circ \{S_2\})[X \mapsto F] = \{(G_3[X \mapsto F], c'_3[X \mapsto F], c'_4[X \mapsto F])\}$
- $\{S_1\}[X \mapsto F] = (G'_1[X \mapsto F], c_1[X \mapsto F], c_1[X \mapsto F])$
- $\{S_2\}[X \mapsto F] = (G'_2[X \mapsto F], c'_2[X \mapsto F], c_2[X \mapsto F])$

Since $\text{pullback}(\mathbf{G}, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}'_2, \mathbf{c}'_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$, by Lemma B.68, we get that

$$\begin{aligned} \text{pullback}(\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}'_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}'_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}])) = \\ (\mathbf{G}_3[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_3[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_4[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_5[\mathbf{X} \mapsto \mathbf{F}]) \end{aligned}$$

Thus, we get that

- Since $\mathbf{c}_3; \mathbf{c}_1 = \mathbf{c}'_3$, by Lemma B.66, we get that $\mathbf{c}_3[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}'_3[\mathbf{X} \mapsto \mathbf{F}]$.
- Since $\mathbf{c}_4; \mathbf{c}_2 = \mathbf{c}'_4$, by Lemma B.66, we get that $\mathbf{c}_4[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}'_4[\mathbf{X} \mapsto \mathbf{F}]$.
- Thus, we get that $\{\mathbf{S}_1\}[\mathbf{X} \mapsto \mathbf{F}]; \{\mathbf{S}_2\}[\mathbf{X} \mapsto \mathbf{F}] = \{(\mathbf{G}_3[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}'_3[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}'_4[\mathbf{X} \mapsto \mathbf{F}])\}$.

Therefore, the result follows immediately. □

Lemma B.51 *If $\varepsilon_1 : \mathbf{G}_1 \sim \mathbf{G}$, $\varepsilon_2 : \mathbf{G} \sim \mathbf{G}_2$ and $(n, \rho) \in \mathcal{D}[\Delta]$, then $\rho_i(\varepsilon_1 \circledast \varepsilon_2) = \rho_i(\varepsilon_1) \circledast \rho_i(\varepsilon_2)$.*

PROOF. We are required to prove that $\rho_i(\varepsilon_1 \circledast \varepsilon_2) = \rho_i(\varepsilon_1) \circledast \rho_i(\varepsilon_2)$. We proceed by proving that for all $\mathbf{S}_1 \in \varepsilon_1$ and $\mathbf{S}_2 \in \varepsilon_2$, $\rho_i(\{\mathbf{S}_1\} \circledast \{\mathbf{S}_2\}) = \rho_i(\{\mathbf{S}_1\}) \circledast \rho_i(\{\mathbf{S}_2\})$. This result follows by Lemma B.70. Thus, the whole results holds. □

Lemma B.52 *If*

- $\varepsilon_1 : \mathbf{G}_1 \sim \rho_1(\mathbf{G})$
- $\varepsilon_2 : \mathbf{G}_2 \sim \rho_2(\mathbf{G})$
- $\varepsilon : \mathbf{G} \sim \mathbf{G}'$
- $(n, \rho) \in \mathcal{D}[\Delta]$
- $\forall \mathbf{S} \in \varepsilon, \varepsilon_1 \circledast \rho_1(\{\mathbf{S}\}) \iff \varepsilon_2 \circledast \rho_2(\{\mathbf{S}\})$

then $\varepsilon_1 \circledast \rho_1(\varepsilon) \iff \varepsilon_2 \circledast \rho_2(\varepsilon)$.

PROOF. Let us first prove that if $\varepsilon_1 \circledast \rho_1(\varepsilon) \implies \varepsilon_2 \circledast \rho_2(\varepsilon)$ (the other way around is similar). Or what is the same, we are required to prove that if $\varepsilon_1 \circledast \rho_1(\varepsilon)$ fails, then $\varepsilon_2 \circledast \rho_2(\varepsilon)$ fails, and if $\varepsilon_1 \circledast \rho_1(\varepsilon)$ succeeds, then $\varepsilon_2 \circledast \rho_2(\varepsilon)$ succeeds. We know that $\forall \mathbf{S} \in \varepsilon$, if $\varepsilon_1 \circledast \rho_1(\{\mathbf{S}\})$ fails, then $\varepsilon_2 \circledast \rho_2(\{\mathbf{S}\})$ fails, and if $\varepsilon_1 \circledast \rho_1(\{\mathbf{S}\})$ succeeds, then $\varepsilon_2 \circledast \rho_2(\{\mathbf{S}\})$ succeeds.

If $\varepsilon_1 \circledast \rho_1(\varepsilon)$ fails, then we know that $\forall \mathbf{S} \in \varepsilon, \varepsilon_1 \circledast \rho_1(\{\mathbf{S}\})$ fails; thus $\forall \mathbf{S} \in \varepsilon$, we get that $\varepsilon_2 \circledast \rho_2(\{\mathbf{S}\})$ fails, and as conclusion $\varepsilon_2 \circledast \rho_2(\varepsilon)$ fails, and the result holds.

If $\varepsilon_1 \circledast \rho_1(\varepsilon)$ succeeds, then we know that $\exists \mathbf{S} \in \varepsilon, \varepsilon_1 \circledast \rho_1(\{\mathbf{S}\})$ succeeds; thus, we get that $\varepsilon_2 \circledast \rho_2(\{\mathbf{S}\})$ succeeds, and as conclusion $\varepsilon_2 \circledast \rho_2(\varepsilon)$ succeeds, and the result holds. □

Lemma B.53 $\Delta, \mathbf{X} \vdash \mathbf{G}$ if and only if $\Delta \vdash \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}]$.

PROOF. Directly by the definition of $\Delta, \mathbf{X} \vdash \mathbf{G}$. \square

Lemma B.54 If $\Delta \vdash \mathbf{F}$ then $\Delta, \mathbf{X} \vdash \mathbf{\Gamma}$ if and only if $\Delta \vdash \mathbf{\Gamma}[\mathbf{X} \mapsto \mathbf{F}]$.

PROOF. Then, we have to prove that for all $x : \mathbf{G} \in \mathbf{\Gamma}$, $\Delta, \mathbf{X} \vdash \mathbf{G}$ if and only if $\Delta \vdash \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}]$, which follows immediately by Lemma B.53. \square

Lemma B.55 If $\Delta \vdash \mathbf{F}'$ then $\Delta, \mathbf{X}; \mathbf{\Gamma} \vdash \mathbf{t} : \mathbf{G}$ if and only if $\Delta; \mathbf{\Gamma}[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{t}[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}[\mathbf{F}'/\mathbf{X}]$.

PROOF. We know that $\Delta \vdash \mathbf{F}'$. We are required to prove that $\Delta, \mathbf{X}; \mathbf{\Gamma} \vdash \mathbf{t} : \mathbf{G}$ if and only if $\Delta; \mathbf{\Gamma}[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{t}[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}[\mathbf{F}'/\mathbf{X}]$.

We first prove that if $\Delta \vdash \mathbf{F}'$ and $\Delta, \mathbf{X}; \mathbf{\Gamma} \vdash \mathbf{t} : \mathbf{G}$, then $\Delta; \mathbf{\Gamma}[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{t}[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}[\mathbf{F}'/\mathbf{X}]$. We proceed by induction on $\Delta, \mathbf{X}; \mathbf{\Gamma} \vdash \mathbf{t} : \mathbf{G}$.

Case (Gb). We know that $\mathbf{t} = \mathbf{b}$, $\mathbf{G} = \mathbf{B}$, and

$$(Gb) \frac{\theta(\mathbf{b}) = \mathbf{B} \quad \Delta, \mathbf{X} \vdash \mathbf{\Gamma}}{\Delta, \mathbf{X}; \mathbf{\Gamma} \vdash \mathbf{b} : \mathbf{B}}$$

Also, we know that $\mathbf{t}[\mathbf{X} \mapsto \mathbf{F}'] = \mathbf{b}[\mathbf{X} \mapsto \mathbf{F}'] = \mathbf{b}$. Therefore, we are required to prove that $\Delta; \mathbf{\Gamma}[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{b} : \mathbf{B}$, which follows immediately by rule (Gb) and Lemma B.54.

Case (Gx). We know that $\mathbf{t} = \mathbf{y}$, and

$$(Gx) \frac{\mathbf{y} : \mathbf{G} \in \mathbf{\Gamma} \quad \Delta, \mathbf{X} \vdash \mathbf{\Gamma}}{\Delta, \mathbf{X}; \mathbf{\Gamma} \vdash \mathbf{y} : \mathbf{G}}$$

Therefore, we know that $\mathbf{t}[\mathbf{X} \mapsto \mathbf{F}'] = \mathbf{y}[\mathbf{X} \mapsto \mathbf{F}'] = \mathbf{y}$. Therefore, we are required to prove that $\Delta; \mathbf{\Gamma}[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{y} : \mathbf{G}[\mathbf{F}'/\mathbf{X}]$, which follows immediately by (Gx) rule, and since $\mathbf{y} : \mathbf{G} \in \mathbf{\Gamma}$, then $\mathbf{x} : \mathbf{G}[\mathbf{F}'/\mathbf{X}] \in \mathbf{\Gamma}[\mathbf{X} \mapsto \mathbf{F}']$ and Lemma B.54.

Case (Gλ). We know that $\mathbf{t} = \lambda \mathbf{y} : \mathbf{G}_1. \mathbf{t}'$, $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$, and

$$(G\lambda) \frac{\Delta, \mathbf{X}; \mathbf{\Gamma}, \mathbf{y} : \mathbf{G}_1 \vdash \mathbf{t}' : \mathbf{G}_2}{\Delta, \mathbf{X}; \mathbf{\Gamma} \vdash (\lambda \mathbf{y} : \mathbf{G}_1. \mathbf{t}') : \mathbf{G}_1 \rightarrow \mathbf{G}_2}$$

We know that $(\lambda \mathbf{y} : \mathbf{G}_1. \mathbf{t}')[\mathbf{X} \mapsto \mathbf{F}'] = \lambda \mathbf{y} : \mathbf{G}_1[\mathbf{F}'/\mathbf{X}]. (\mathbf{t}'[\mathbf{X} \mapsto \mathbf{F}'])$. We are required to prove that $\Delta; \mathbf{\Gamma}[\mathbf{X} \mapsto \mathbf{F}'] \vdash \lambda \mathbf{y} : \mathbf{G}_1[\mathbf{F}'/\mathbf{X}]. (\mathbf{t}'[\mathbf{X} \mapsto \mathbf{F}']) : \mathbf{G}_1[\mathbf{F}'/\mathbf{X}] \rightarrow \mathbf{G}_2[\mathbf{F}'/\mathbf{X}]$. Or what is the same, by rule (Gλ), we are required to prove $\Delta; \mathbf{\Gamma}, \mathbf{y} : \mathbf{G}_1[\mathbf{F}'/\mathbf{X}] \vdash \mathbf{t}'[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}_2[\mathbf{F}'/\mathbf{X}]$. By the induction hypothesis on $\Delta, \mathbf{X}; \mathbf{\Gamma}, \mathbf{y} : \mathbf{G}_1 \vdash \mathbf{t}' : \mathbf{G}_2$, the result follows immediately.

Case (Gpair). We know that $\mathbf{t} = \langle \mathbf{s}_1, \mathbf{s}_2 \rangle$, $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$, and

$$(Gpair) \frac{\Delta, \mathbf{X}; \mathbf{\Gamma} \vdash \mathbf{s}_1 : \mathbf{G}_1 \quad \Delta, \mathbf{X}; \mathbf{\Gamma} \vdash \mathbf{s}_2 : \mathbf{G}_2}{\Delta, \mathbf{X}; \mathbf{\Gamma} \vdash \langle \mathbf{s}_1, \mathbf{s}_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2}$$

We know that $(\langle \mathbf{s}_1, \mathbf{s}_2 \rangle)[\mathbf{X} \mapsto \mathbf{F}'] = \langle \mathbf{s}_1[\mathbf{X} \mapsto \mathbf{F}'], \mathbf{s}_2[\mathbf{X} \mapsto \mathbf{F}'] \rangle$. We are required to prove that $\Delta; \mathbf{\Gamma}[\mathbf{X} \mapsto \mathbf{F}'] \vdash \langle \mathbf{s}_1[\mathbf{X} \mapsto \mathbf{F}'], \mathbf{s}_2[\mathbf{X} \mapsto \mathbf{F}'] \rangle : \mathbf{G}_1[\mathbf{F}'/\mathbf{X}] \times \mathbf{G}_2[\mathbf{F}'/\mathbf{X}]$. Or what is the same, by rule (Gpair), we are required to prove that $\Delta; \mathbf{\Gamma}[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{s}_1[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}_1[\mathbf{F}'/\mathbf{X}]$ and $\Delta; \mathbf{\Gamma}[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{s}_2[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}_2[\mathbf{F}'/\mathbf{X}]$. By the induction hypothesis on $\Delta, \mathbf{X}; \mathbf{\Gamma} \vdash \mathbf{s}_1 : \mathbf{G}_1$ and $\Delta, \mathbf{X}; \mathbf{\Gamma} \vdash \mathbf{s}_2 : \mathbf{G}_2$, the result follows immediately.

Case (G Λ). We know that $\mathbf{t} = \Lambda \mathbf{Y}. \mathbf{t}'$, $\mathbf{G} = \forall \mathbf{Y}. \mathbf{G}''$, and

$$(\mathbf{G}\Lambda) \frac{\Delta, \mathbf{X}, \mathbf{Y}; \Gamma \vdash \mathbf{t}' : \mathbf{G}'' \quad \Delta, \mathbf{X} \vdash \Gamma}{\Delta, \mathbf{X}; \Gamma \vdash (\Lambda \mathbf{Y}. \mathbf{t}') : \forall \mathbf{Y}. \mathbf{G}''}$$

We know that $(\Lambda \mathbf{Y}. \mathbf{t}')[\mathbf{X} \mapsto \mathbf{F}'] = \Lambda \mathbf{Y}. (\mathbf{t}'[\mathbf{X} \mapsto \mathbf{F}'])$. We are required to prove that $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \Lambda \mathbf{Y}. (\mathbf{t}'[\mathbf{X} \mapsto \mathbf{F}']) : \forall \mathbf{Y}. (\mathbf{G}''[\mathbf{F}'/\mathbf{X}])$. Or what is the same, by rule (G Λ), we are required to prove that $\Delta, \mathbf{Y}; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{t}'[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}''[\mathbf{F}'/\mathbf{X}]$. By the induction hypothesis on $\Delta, \mathbf{X}, \mathbf{Y}; \Gamma \vdash \mathbf{t}' : \mathbf{G}''$, the result follows immediately.

Case (Gasc). We know that $\mathbf{t} = \varepsilon \mathbf{s} :: \mathbf{G}$, and

$$(\mathbf{G}\text{asc}) \frac{\Delta, \mathbf{X}; \Gamma \vdash \mathbf{s} : \mathbf{G}'' \quad \varepsilon : \mathbf{G}'' \sim \mathbf{G}}{\Delta, \mathbf{X}; \Gamma \vdash \varepsilon \mathbf{s} :: \mathbf{G} : \mathbf{G}}$$

We know that $(\varepsilon \mathbf{s} :: \mathbf{G})[\mathbf{X} \mapsto \mathbf{F}'] = (\varepsilon[\mathbf{X} \mapsto \mathbf{F}'] (\mathbf{s}[\mathbf{X} \mapsto \mathbf{F}']) :: \mathbf{G}[\mathbf{F}'/\mathbf{X}])$. Thus, we are required to prove that $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \varepsilon[\mathbf{X} \mapsto \mathbf{F}'] (\mathbf{s}[\mathbf{X} \mapsto \mathbf{F}']) :: \mathbf{G}[\mathbf{F}'/\mathbf{X}] : \mathbf{G}[\mathbf{F}'/\mathbf{X}]$. Or what is the same, by (Gasc) rule, we are required to prove that $\varepsilon[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}''[\mathbf{F}'/\mathbf{X}] \sim \mathbf{G}''[\mathbf{F}'/\mathbf{X}]$, which follows by Lemma B.58, and $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash (\mathbf{s}[\mathbf{X} \mapsto \mathbf{F}']) : \mathbf{G}''[\mathbf{F}'/\mathbf{X}]$. By the induction hypothesis on $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{s} : \mathbf{G}''$, the result follows immediately.

Case (Gapp). We know that $\mathbf{t} = \mathbf{t}_1 \mathbf{t}_2$ and $\mathbf{G} = \mathbf{G}_2$, and

$$(\mathbf{G}\text{app}) \frac{\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \quad \Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}_2 : \mathbf{G}_1}{\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}_1 \mathbf{t}_2 : \mathbf{G}_2}$$

We know that $(\mathbf{t}_1 \mathbf{t}_2)[\mathbf{X} \mapsto \mathbf{F}'] = (\mathbf{t}_1[\mathbf{X} \mapsto \mathbf{F}']) (\mathbf{t}_2[\mathbf{X} \mapsto \mathbf{F}'])$. Therefore, we are required to prove that $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash (\mathbf{t}_1[\mathbf{X} \mapsto \mathbf{F}']) (\mathbf{t}_2[\mathbf{X} \mapsto \mathbf{F}']) : \mathbf{G}_2[\mathbf{F}'/\mathbf{X}]$. Or what is the same, by rule (Gapp), we are required to prove that $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{t}_1[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}_1[\mathbf{F}'/\mathbf{X}] \rightarrow \mathbf{G}_2[\mathbf{F}'/\mathbf{X}]$ and $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{t}_2[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}_1[\mathbf{F}'/\mathbf{X}]$, which follow immediately by the induction hypothesis on $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}_2 : \mathbf{G}_1$.

Case (Gpairi). We know that $\mathbf{t} = \pi_i(\mathbf{t}')$ and $\mathbf{G} = \mathbf{G}_i$, and

$$(\mathbf{G}\text{pair}i) \frac{\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}' : \mathbf{G}_1 \times \mathbf{G}_2}{\Delta, \mathbf{X}; \Gamma \vdash \pi_i(\mathbf{t}') : \mathbf{G}_i}$$

We know that $(\pi_i(\mathbf{t}'))[\mathbf{X} \mapsto \mathbf{F}'] = \pi_i((\mathbf{t}'[\mathbf{X} \mapsto \mathbf{F}']))$. Therefore, we are required to prove that $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \pi_i((\mathbf{t}'[\mathbf{X} \mapsto \mathbf{F}'])) : \mathbf{G}_i[\mathbf{F}'/\mathbf{X}]$. Or what is the same, by rule (Gpairi), we are required to prove that $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{t}'[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}_1[\mathbf{F}'/\mathbf{X}] \times \mathbf{G}_2[\mathbf{F}'/\mathbf{X}]$, which follow immediately by the induction hypothesis on $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}' : \mathbf{G}_1 \times \mathbf{G}_2$.

Case (GappG). We know that $\mathbf{t} = \mathbf{t}'[\mathbf{F}]$ and $\mathbf{G} = \mathbf{G}''[\mathbf{F}/\mathbf{X}]$, and

$$(\mathbf{G}\text{app}G) \frac{\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}' : \forall \mathbf{X}. \mathbf{G}'' \quad \Delta, \mathbf{X} \vdash \mathbf{F}}{\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}'[\mathbf{F}] : \mathbf{G}''[\mathbf{F}/\mathbf{X}]}$$

We know that $(\mathbf{t}'[\mathbf{F}])[\mathbf{X} \mapsto \mathbf{F}'] = (\mathbf{t}'[\mathbf{X} \mapsto \mathbf{F}'])[\mathbf{F}[\mathbf{F}'/\mathbf{X}]]$. Therefore, we are required to prove that $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash (\mathbf{t}'[\mathbf{X} \mapsto \mathbf{F}'])[\mathbf{F}[\mathbf{F}'/\mathbf{X}]] : \mathbf{G}''[\mathbf{F}[\mathbf{F}'/\mathbf{X}]/\mathbf{X}]$. Or what is the same, by the rule (GappG), we are required to prove that $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{t}'[\mathbf{X} \mapsto \mathbf{F}'] : \forall \mathbf{X}. (\mathbf{G}''[\mathbf{F}'/\mathbf{X}])$, which follows immediately by the induction hypotheses on $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t}' : \forall \mathbf{X}. \mathbf{G}''$.

Next, we prove the other way around. We are required to prove that if $\Delta \vdash \mathbf{F}'$ and $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{t}[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}[\mathbf{F}'/\mathbf{X}]$, then $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{t} : \mathbf{G}$.

Case (b). We know that $\mathbf{t} = \mathbf{b}$, and $\mathbf{t}[\mathbf{X} \mapsto \mathbf{F}'] = \mathbf{b}[\mathbf{X} \mapsto \mathbf{F}'] = \mathbf{b}$. Also, we know that

$$(\mathbf{Gb}) \frac{\theta(\mathbf{b}) = \mathbf{B} \quad \Delta \vdash \Gamma[\mathbf{X} \mapsto \mathbf{F}']}{\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{b} : \mathbf{B}}$$

Thus, we know that $\mathbf{G} = \mathbf{B}$ and we are required to prove that $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{b} : \mathbf{B}$, which follows immediately by rule (Gb) and Lemma B.54.

Case (y). We know that $\mathbf{t} = \mathbf{y}$, and $\mathbf{t}[\mathbf{X} \mapsto \mathbf{F}'] = \mathbf{y}[\mathbf{X} \mapsto \mathbf{F}'] = \mathbf{y}$. Thus, we know that

$$(\mathbf{Gx}) \frac{\mathbf{y} : \mathbf{G}[\mathbf{F}'/\mathbf{X}] \in \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \quad \Delta \vdash \Gamma[\mathbf{X} \mapsto \mathbf{F}']}{\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{y} : \mathbf{G}[\mathbf{F}'/\mathbf{X}]}$$

Therefore, we are required to prove that $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{y} : \mathbf{G}$, which follows immediately by (Gx) rule, Lemma B.54, and since $\mathbf{y} : \mathbf{G}[\mathbf{F}'/\mathbf{X}] \in \Gamma[\mathbf{X} \mapsto \mathbf{F}']$, then $\mathbf{y} : \mathbf{G} \in \Gamma$.

Case (λ). We know that $\mathbf{t} = \lambda \mathbf{y} : \mathbf{G}_1.t'$, and $(\lambda \mathbf{y} : \mathbf{G}_1.t')[\mathbf{X} \mapsto \mathbf{F}'] = \lambda \mathbf{y} : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}'].(t'[\mathbf{X} \mapsto \mathbf{F}'])$. Thus, we also know that

$$(\mathbf{G}\lambda) \frac{\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'], \mathbf{y} : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}'] \vdash t'[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}']}{\Delta; \Gamma \vdash (\lambda \mathbf{y} : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}']).(t'[\mathbf{X} \mapsto \mathbf{F}']) : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}'] \rightarrow \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}']}$$

where $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$. We re required to prove that $\Delta, \mathbf{X}; \Gamma \vdash \lambda \mathbf{y} : \mathbf{G}_1.t' : \mathbf{G}_1 \rightarrow \mathbf{G}_2$. Or what is the same, by rule (G λ), we are required to prove that $\Delta, \mathbf{X}; \Gamma, \mathbf{y} : \mathbf{G}_1 \vdash t' : \mathbf{G}_2$. By the induction hypothesis on t' , with $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'], \mathbf{y} : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}'] \vdash t'[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}']$, the result follows immediately.

Case (pair). We know that $\mathbf{t} = \langle \mathbf{s}_1, \mathbf{s}_2 \rangle$, and $(\langle \mathbf{s}_1, \mathbf{s}_2 \rangle)[\mathbf{X} \mapsto \mathbf{F}'] = \langle \mathbf{s}_1[\mathbf{X} \mapsto \mathbf{F}'], \mathbf{s}_2[\mathbf{X} \mapsto \mathbf{F}'] \rangle$. Thus, we know that

$$(\mathbf{Gpair}) \frac{\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{s}_1[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}'] \quad \Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{s}_2[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}']}{\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \langle \mathbf{s}_1[\mathbf{X} \mapsto \mathbf{F}'], \mathbf{s}_2[\mathbf{X} \mapsto \mathbf{F}'] \rangle : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}'] \times \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}']}$$

where $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$. We are required to prove that $\Delta, \mathbf{X}; \Gamma \vdash \langle \mathbf{s}_1, \mathbf{s}_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2$. Or what is the same, by rule (Gpair), we are required to prove that $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{s}_1 : \mathbf{G}_1$ and $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{s}_2 : \mathbf{G}_2$. By the induction hypothesis on \mathbf{s}_1 and \mathbf{s}_2 , with $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{s}_1[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}']$ and $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{s}_2[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}']$, the result follows immediately.

Case (Λ). We know that $\mathbf{t} = \Lambda \mathbf{Y}.t'$, and $(\Lambda \mathbf{Y}.t')[\mathbf{X} \mapsto \mathbf{F}'] = \Lambda \mathbf{Y}.(t'[\mathbf{X} \mapsto \mathbf{F}'])$. Thus, we know that

$$(\mathbf{G}\Lambda) \frac{\Delta, \mathbf{Y}; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash t'[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}''[\mathbf{X} \mapsto \mathbf{F}'] \quad \Delta \vdash \Gamma}{\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \Lambda \mathbf{Y}.(t'[\mathbf{X} \mapsto \mathbf{F}']) : \forall \mathbf{Y}.(\mathbf{G}''[\mathbf{X} \mapsto \mathbf{F}'])}$$

where $\mathbf{G} = \forall \mathbf{Y}. \mathbf{G}''$. We are required to prove that $\Delta, \mathbf{X}; \Gamma \vdash \Lambda \mathbf{Y}.t' : \forall \mathbf{Y}. \mathbf{G}''$. Or what is the same, by rule (G Λ), we are required to prove that $\Delta, \mathbf{Y}; \Gamma \vdash t' : \mathbf{G}''$. By the induction hypothesis on t' , with $\Delta, \mathbf{Y}; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash t'[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}''[\mathbf{X} \mapsto \mathbf{F}']$, the result follows immediately.

Case (asc). We know that $\mathbf{t} = \varepsilon \mathbf{s} :: \mathbf{G}$, and $(\varepsilon \mathbf{s} :: \mathbf{G})[\mathbf{X} \mapsto \mathbf{F}'] = (\varepsilon[\mathbf{X} \mapsto \mathbf{F}'](\mathbf{s}[\mathbf{X} \mapsto \mathbf{F}'])) :: \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}']$. Thus, we know that

$$(\mathbf{Gasc}) \frac{\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{s}[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}''[\mathbf{X} \mapsto \mathbf{F}'] \quad \varepsilon[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}''[\mathbf{X} \mapsto \mathbf{F}'] \sim \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}']}{\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \varepsilon[\mathbf{X} \mapsto \mathbf{F}'](\mathbf{s}[\mathbf{X} \mapsto \mathbf{F}']) :: \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}']}$$

Thus, we are required to prove that $\Delta, \mathbf{X}; \Gamma \vdash (\varepsilon \mathbf{s} :: \mathbf{G}) : \mathbf{G}$. Or what is the same, by (Gasc) rule, we are required to prove that $\varepsilon : \mathbf{G}'' \sim \mathbf{G}$, which follows by Lemma B.58, and $\Delta, \mathbf{X}; \Gamma \vdash \mathbf{s} : \mathbf{G}''$. By the induction hypothesis on \mathbf{s} , with $\Delta; \Gamma[\mathbf{X} \mapsto \mathbf{F}'] \vdash \mathbf{s}[\mathbf{X} \mapsto \mathbf{F}'] : \mathbf{G}''[\mathbf{X} \mapsto \mathbf{F}']$, the result follows immediately.

Case (app). We know that $t = t_1 t_2$, and $(t_1 t_2)[X \mapsto F'] = (t_1[X \mapsto F'])(t_2[X \mapsto F'])$. Thus, we know that

$$(\mathbf{G}_{\text{app}}) \frac{\Delta; \Gamma[X \mapsto F'] \vdash t_1[X \mapsto F'] : G_1[X \mapsto F'] \rightarrow G_2[X \mapsto F'] \quad \Delta; \Gamma[X \mapsto F'] \vdash t_2[X \mapsto F'] : G_1[X \mapsto F']}{\Delta; \Gamma[X \mapsto F'] \vdash (t_1[X \mapsto F'])(t_2[X \mapsto F']) : G_2[X \mapsto F']}$$

where $G = G_2$. Therefore, we are required to prove that $\Delta, X; \Gamma \vdash t_1 t_2 : G_2$. Or what is the same, by rule $(\mathbf{G}_{\text{app}})$, we are required to prove that $\Delta, X; \Gamma \vdash t_1 : G_1 \rightarrow G_2$ and $\Delta, X; \Gamma \vdash t_2 : G_1$, which follow immediately by the induction hypothesis on t_1 and t_2 , with $\Delta; \Gamma[X \mapsto F'] \vdash t_1[X \mapsto F'] : G_1[X \mapsto F'] \rightarrow G_2[X \mapsto F']$ and $\Delta; \Gamma[X \mapsto F'] \vdash t_2[X \mapsto F'] : G_1[X \mapsto F']$.

Case (pairi). We know that $t = \pi_i(t')$, and $(\pi_i(t'))[X \mapsto F'] = \pi_i(t'[X \mapsto F'])$. Thus, we know that

$$(\mathbf{G}_{\text{pairi}}) \frac{\Delta; \Gamma[X \mapsto F'] \vdash t'[X \mapsto F'] : G_1[X \mapsto F'] \times G_2[X \mapsto F']}{\Delta; \Gamma[X \mapsto F'] \vdash \pi_i(t'[X \mapsto F']) : G_i[X \mapsto F']}$$

where $G = G_i$. Therefore, we are required to prove that $\Delta, X; \Gamma \vdash \pi_i(t') : G_i$. Or what is the same, by rule $(\mathbf{G}_{\text{pairi}})$, we are required to prove that $\Delta, X; \Gamma \vdash t' : G_1 \times G_2$, which follow immediately by the induction hypothesis on t' , with $\Delta; \Gamma[X \mapsto F'] \vdash t'[X \mapsto F'] : G_1[X \mapsto F'] \times G_2[X \mapsto F']$.

Case (appG). We know that $t = t' [F]$, and $(t' [F])[X \mapsto F'] = (t'[X \mapsto F']) [F[X \mapsto F']]$. Thus, we know that

$$(\mathbf{G}_{\text{appG}}) \frac{\Delta; \Gamma[X \mapsto F'] \vdash t'[X \mapsto F'] : \forall Y. G''[X \mapsto F'] \quad \Delta \vdash F[X \mapsto F']}{\Delta; \Gamma[X \mapsto F'] \vdash (t'[X \mapsto F']) [F[X \mapsto F']] : G''[F[X \mapsto F']/X]}$$

where $G = G''[F/X]$. Therefore, we are required to prove that $\Delta, X; \Gamma \vdash t' [F] : G''[F/X]$. Or what is the same, by the rule $(\mathbf{G}_{\text{appG}})$, we are required to prove that $\Delta, X; \Gamma \vdash t' : \forall Y. G''$, which follows immediately by the induction hypotheses on t' , with $\Delta; \Gamma[X \mapsto F'] \vdash t'[X \mapsto F'] : \forall Y. G''[X \mapsto F']$.

□

Lemma B.56 *If $\Delta; \Gamma \vdash v : G'$ then $\Delta; \Gamma, x : G' \vdash t : G$ if and only if $\Delta; \Gamma \vdash t[v/x] : G$.*

PROOF. We first prove that if $\Delta; \Gamma \vdash v : G'$ and $\Delta; \Gamma, x : G' \vdash t : G$, then $\Delta; \Gamma \vdash t[v/x] : G$. We proceed by induction on $\Delta; \Gamma \vdash t : G$.

Case (Gb). We know that $t = b$, $G = B$, and

$$(\mathbf{G}_b) \frac{\theta(b) = B \quad \Delta \vdash \Gamma, x : G'}{\Delta; \Gamma, x : G' \vdash b : B}$$

Also, we know that $t[v/x] = b[v/x] = b$. Therefore, we are required to prove that $\Delta; \Gamma \vdash b : B$, which follows immediately.

Case (Gx). We know that $t = y$, and

$$(\mathbf{G}_x) \frac{y : G \in \Gamma \quad \Delta \vdash \Gamma, x : G'}{\Delta; \Gamma, x : G' \vdash y : G}$$

We have the following cases.

- $y = x$. We know that $t[v/x] = x[v/x] = v$. Then, we know that $G' = G$. Therefore, we are required to prove that $\Delta; \Gamma \vdash v : G$, which follows immediately by the premise.

- $y \neq x$. We know that $\mathbf{t}[v/x] = \mathbf{y}_{v/x} = \mathbf{y}$. Therefore, we are required to prove that $\Delta; \Gamma \vdash \mathbf{y} : \mathbf{G}$, which follows immediately ($\mathbf{y} : \mathbf{G} \in \Gamma$).

Case ($\mathbf{G}\lambda$). We know that $\mathbf{t} = \lambda \mathbf{y} : \mathbf{G}_1.t'$, $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$, and

$$(\mathbf{G}\lambda) \frac{\Delta; \Gamma, \mathbf{x} : \mathbf{G}', \mathbf{y} : \mathbf{G}_1 \vdash t' : \mathbf{G}_2}{\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash (\lambda \mathbf{y} : \mathbf{G}_1.t') : \mathbf{G}_1 \rightarrow \mathbf{G}_2}$$

We know that $(\lambda \mathbf{y} : \mathbf{G}_1.t')[v/x] = \lambda \mathbf{y} : \mathbf{G}_1.(t'[v/x])$. We are required to prove that $\Delta; \Gamma \vdash \lambda \mathbf{y} : \mathbf{G}_1.(t'[v/x]) : \mathbf{G}_1 \rightarrow \mathbf{G}_2$. Or what is the same, by rule ($\mathbf{G}\lambda$), we are required to prove $\Delta; \Gamma, \mathbf{y} : \mathbf{G}_1 \vdash t'[v/x] : \mathbf{G}_2$. By the induction hypothesis on $\Delta; \Gamma, \mathbf{x} : \mathbf{G}', \mathbf{y} : \mathbf{G}_1 \vdash t' : \mathbf{G}_2$, we get that $\Delta; \Gamma, \mathbf{y} : \mathbf{G}_1 \vdash t'[v/x] : \mathbf{G}_2$, and the result follows immediately.

Case ($\mathbf{G}\text{pair}$). We know that $\mathbf{t} = \langle \mathbf{s}_1, \mathbf{s}_2 \rangle$, $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$, and

$$(\mathbf{G}\text{pair}) \frac{\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{s}_1 : \mathbf{G}_1 \quad \Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{s}_2 : \mathbf{G}_2}{\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \langle \mathbf{s}_1, \mathbf{s}_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2}$$

We know that $(\langle \mathbf{s}_1, \mathbf{s}_2 \rangle)[v/x] = \langle \mathbf{s}_1[v/x], \mathbf{s}_2[v/x] \rangle$. We are required to prove that $\Delta; \Gamma \vdash \langle \mathbf{s}_1[v/x], \mathbf{s}_2[v/x] \rangle : \mathbf{G}_1 \times \mathbf{G}_2$. Or what is the same, by rule ($\mathbf{G}\text{pair}$), we are required to prove that $\Delta; \Gamma \vdash \mathbf{s}_1[v/x] : \mathbf{G}_1$ and $\Delta; \Gamma \vdash \mathbf{s}_2[v/x] : \mathbf{G}_2$. By the induction hypothesis on $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{s}_1 : \mathbf{G}_1$ and $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{s}_2 : \mathbf{G}_2$, the result follows immediately.

Case ($\mathbf{G}\Lambda$). We know that $\mathbf{t} = \Lambda \mathbf{X}.t'$, $\mathbf{G} = \forall \mathbf{X}.\mathbf{G}''$, and

$$(\mathbf{G}\Lambda) \frac{\Delta, \mathbf{X}; \Gamma, \mathbf{x} : \mathbf{G}' \vdash t' : \mathbf{G}'' \quad \Delta \vdash \Gamma, \mathbf{x} : \mathbf{G}'}{\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash (\Lambda \mathbf{X}.t') : \forall \mathbf{X}.\mathbf{G}''}$$

We know that $(\Lambda \mathbf{X}.t')[v/x] = \Lambda \mathbf{X}.(t'[v/x])$. We are required to prove that $\Delta; \Gamma \vdash \Lambda \mathbf{X}.(t'[v/x]) : \forall \mathbf{X}.\mathbf{G}''$. Or what is the same, by rule ($\mathbf{G}\Lambda$), we are required to prove that $\Delta, \mathbf{X}; \Gamma \vdash t'[v/x] : \mathbf{G}''$. By the induction hypothesis on $\Delta, \mathbf{X}; \Gamma, \mathbf{x} : \mathbf{G}' \vdash t' : \mathbf{G}''$, the result follows immediately.

Case ($\mathbf{G}\text{asc}$). We know that $\mathbf{t} = \varepsilon \mathbf{s} :: \mathbf{G}$, and

$$(\mathbf{G}\text{asc}) \frac{\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{s} : \mathbf{G}'' \quad \varepsilon : \mathbf{G}'' \sim \mathbf{G}}{\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \varepsilon \mathbf{s} :: \mathbf{G} : \mathbf{G}}$$

We know that $(\varepsilon \mathbf{s} :: \mathbf{G})[v/x] = (\varepsilon(\mathbf{s}[v/x]) :: \mathbf{G})$. Thus, we are required to prove that $\Delta; \Gamma \vdash (\varepsilon(\mathbf{s}[v/x]) :: \mathbf{G}) : \mathbf{G}$. Or what is the same, by ($\mathbf{G}\text{asc}$) rule, we are required to prove that $\Delta; \Gamma \vdash (\mathbf{s}[v/x]) : \mathbf{G}''$. By the induction hypothesis on $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{s} : \mathbf{G}''$, the result follows immediately.

Case ($\mathbf{G}\text{app}$). We know that $\mathbf{t} = \mathbf{t}_1 \mathbf{t}_2$ and $\mathbf{G} = \mathbf{G}_2$, and

$$(\mathbf{G}\text{app}) \frac{\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{t}_1 : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \quad \Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{t}_2 : \mathbf{G}_1}{\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{t}_1 \mathbf{t}_2 : \mathbf{G}_2}$$

We know that $(\mathbf{t}_1 \mathbf{t}_2)[v/x] = (\mathbf{t}_1[v/x]) (\mathbf{t}_2[v/x])$. Therefore, we are required to prove that $\Delta; \Gamma \vdash (\mathbf{t}_1[v/x]) (\mathbf{t}_2[v/x]) : \mathbf{G}_2$. Or what is the same, by rule ($\mathbf{G}\text{app}$), we are required to prove that $\Delta; \Gamma \vdash \mathbf{t}_1[v/x] : \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\Delta; \Gamma \vdash \mathbf{t}_2[v/x] : \mathbf{G}_1$, which follow immediately by the induction hypothesis on $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{t}_1 : \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{t}_2 : \mathbf{G}_1$.

Case ($\mathbf{G}\text{pair}_i$). We know that $\mathbf{t} = \pi_i(t')$ and $\mathbf{G} = \mathbf{G}_i$, and

$$(\mathbf{G}\text{pair}_i) \frac{\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash t' : \mathbf{G}_1 \times \mathbf{G}_2}{\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \pi_i(t') : \mathbf{G}_i}$$

We know that $(\pi_i(t'))[v/x] = \pi_i((t'[v/x]))$. Therefore, we are required to prove that $\Delta; \Gamma \vdash \pi_i((t'[v/x])) : \mathbf{G}_i$. Or what is the same, by rule ($\mathbf{G}\text{pair}_i$), we are required to prove that $\Delta; \Gamma \vdash t'[v/x] : \mathbf{G}_1 \times \mathbf{G}_2$, which follow immediately by the induction hypothesis on $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash t' : \mathbf{G}_1 \times \mathbf{G}_2$.

Case (\mathbf{G}_{appG}). We know that $\mathbf{t} = \mathbf{t}' [\mathbf{F}]$ and $\mathbf{G} = \mathbf{G}''[\mathbf{F}/\mathbf{X}]$, and

$$(\mathbf{G}_{\text{appG}}) \frac{\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{t}' : \forall \mathbf{X}. \mathbf{G}'' \quad \Delta \vdash \mathbf{F}}{\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{t}' [\mathbf{F}] : \mathbf{G}''[\mathbf{F}/\mathbf{X}]}$$

We know that $(\mathbf{t}' [\mathbf{F}])[\mathbf{v}/\mathbf{x}] = (\mathbf{t}'[\mathbf{v}/\mathbf{x}]) [\mathbf{F}]$. Therefore, we are required to prove that $\Delta; \Gamma \vdash (\mathbf{t}'[\mathbf{v}/\mathbf{x}]) [\mathbf{F}] : \mathbf{G}''[\mathbf{F}/\mathbf{X}]$. Or what is the same, by the rule (\mathbf{G}_{appG}), we are required to prove that $\Delta; \Gamma \vdash \mathbf{t}'[\mathbf{v}/\mathbf{x}] : \forall \mathbf{X}. \mathbf{G}''$, which follows immediately by the induction hypotheses on $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{t}' : \forall \mathbf{X}. \mathbf{G}''$.

Next, we prove the other way around. We are required to prove that if $\Delta; \Gamma \vdash \mathbf{v} : \mathbf{G}'$ and $\Delta; \Gamma \vdash \mathbf{t}[\mathbf{v}/\mathbf{x}] : \mathbf{G}$, then $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{t} : \mathbf{G}$. We proceed by induction on the structure of \mathbf{t} .

Case (\mathbf{b}). We know that $\mathbf{t} = \mathbf{b}$, and $\mathbf{t}[\mathbf{v}/\mathbf{x}] = \mathbf{b}[\mathbf{v}/\mathbf{x}] = \mathbf{b}$. Also, we know that

$$(\mathbf{Gb}) \frac{\theta(\mathbf{b}) = \mathbf{B} \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \mathbf{b} : \mathbf{B}}$$

Thus, we know that $\mathbf{G} = \mathbf{B}$ and we are required to prove that $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{b} : \mathbf{B}$, which follows immediately by rule (\mathbf{Gb}).

Case (\mathbf{y}). We know that $\mathbf{t} = \mathbf{y}$ having the following cases.

- $y = x$. We know that $\mathbf{t}[\mathbf{v}/\mathbf{x}] = \mathbf{x}[\mathbf{v}/\mathbf{x}] = v$. Then, we know that $\mathbf{G}' = \mathbf{G}$. Therefore, we are required to prove that $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{x} : \mathbf{G}'$, which follows immediately by the (\mathbf{Gx}) rule.
- $y \neq x$. We know that $\mathbf{t}[\mathbf{v}/\mathbf{x}] = \mathbf{y}_{\mathbf{v}/\mathbf{x}} = \mathbf{y}$ and

$$(\mathbf{Gx}) \frac{\mathbf{y} : \mathbf{G} \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \mathbf{y} : \mathbf{G}}$$

Therefore, we are required to prove that $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{y} : \mathbf{G}$, which follows immediately by the (\mathbf{Gx}) rule ($\mathbf{y} : \mathbf{G} \in \Gamma$).

Case (λ). We know that $\mathbf{t} = \lambda \mathbf{y} : \mathbf{G}_1. \mathbf{t}'$, and $(\lambda \mathbf{y} : \mathbf{G}_1. \mathbf{t}')[\mathbf{v}/\mathbf{x}] = \lambda \mathbf{y} : \mathbf{G}_1. (\mathbf{t}'[\mathbf{v}/\mathbf{x}])$. Thus, we also know that

$$(\mathbf{G}\lambda) \frac{\Delta; \Gamma, \mathbf{y} : \mathbf{G}_1 \vdash \mathbf{t}'[\mathbf{v}/\mathbf{x}] : \mathbf{G}_2}{\Delta; \Gamma \vdash (\lambda \mathbf{y} : \mathbf{G}_1. (\mathbf{t}'[\mathbf{v}/\mathbf{x}])) : \mathbf{G}_1 \rightarrow \mathbf{G}_2}$$

where $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$. We re required to prove that $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \lambda \mathbf{y} : \mathbf{G}_1. \mathbf{t}' : \mathbf{G}_1 \rightarrow \mathbf{G}_2$. Or what is the same, by rule ($\mathbf{G}\lambda$), we are required to prove $\Delta; \Gamma, \mathbf{x} : \mathbf{G}', \mathbf{y} : \mathbf{G}_1 \vdash \mathbf{t}' : \mathbf{G}_2$. By the induction hypothesis on \mathbf{t}' , with $\Delta; \Gamma, \mathbf{y} : \mathbf{G}_1 \vdash \mathbf{t}'[\mathbf{v}/\mathbf{x}] : \mathbf{G}_2$, we get that $\Delta; \Gamma, \mathbf{x} : \mathbf{G}', \mathbf{y} : \mathbf{G}_1 \vdash \mathbf{t}' : \mathbf{G}_2$, and the result follows immediately.

Case (pair). We know that $\mathbf{t} = \langle \mathbf{s}_1, \mathbf{s}_2 \rangle$, and $(\langle \mathbf{s}_1, \mathbf{s}_2 \rangle)[\mathbf{v}/\mathbf{x}] = \langle \mathbf{s}_1[\mathbf{v}/\mathbf{x}], \mathbf{s}_2[\mathbf{v}/\mathbf{x}] \rangle$. Thus, we know that

$$(\mathbf{G}_{\text{pair}}) \frac{\Delta; \Gamma \vdash \mathbf{s}_1[\mathbf{v}/\mathbf{x}] : \mathbf{G}_1 \quad \Delta; \Gamma \vdash \mathbf{s}_2[\mathbf{v}/\mathbf{x}] : \mathbf{G}_2}{\Delta; \Gamma \vdash \langle \mathbf{s}_1[\mathbf{v}/\mathbf{x}], \mathbf{s}_2[\mathbf{v}/\mathbf{x}] \rangle : \mathbf{G}_1 \times \mathbf{G}_2}$$

where $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$. We are required to prove that $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \langle \mathbf{s}_1, \mathbf{s}_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2$. Or what is the same, by rule (\mathbf{G}_{pair}), we are required to prove that $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{s}_1 : \mathbf{G}_1$ and $\Delta; \Gamma, \mathbf{x} : \mathbf{G}' \vdash \mathbf{s}_2 : \mathbf{G}_2$. By the induction hypothesis on \mathbf{s}_1 and \mathbf{s}_2 , with $\Delta; \Gamma \vdash \mathbf{s}_1[\mathbf{v}/\mathbf{x}] : \mathbf{G}_1$ and $\Delta; \Gamma \vdash \mathbf{s}_2[\mathbf{v}/\mathbf{x}] : \mathbf{G}_2$, the result follows immediately.

Case (Λ). We know that $t = \Lambda X.t'$, and $(\Lambda X.t')[v/x] = \Lambda X.(t'[v/x])$. Thus, we know that

$$(\mathbf{G}\Lambda) \frac{\Delta, X; \Gamma \vdash t'[v/x] : \mathbf{G}'' \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \Lambda X.(t'[v/x]) : \forall X.\mathbf{G}''}$$

where $\mathbf{G} = \forall X.\mathbf{G}''$. We are required to prove that $\Delta; \Gamma, x : \mathbf{G}' \vdash \Lambda X.t' : \forall X.\mathbf{G}''$. Or what is the same, by rule ($\mathbf{G}\Lambda$), we are required to prove that $\Delta, X; \Gamma, x : \mathbf{G}' \vdash t' : \mathbf{G}''$. By the induction hypothesis on t' , with $\Delta, X; \Gamma \vdash t'[v/x] : \mathbf{G}''$, the result follows immediately.

Case (asc). We know that $t = \varepsilon s :: \mathbf{G}$, and $(\varepsilon s :: \mathbf{G})[v/x] = (\varepsilon (s[v/x])) :: \mathbf{G}$. Thus, we know that

$$(\mathbf{G}\text{asc}) \frac{\Delta; \Gamma \vdash s[v/x] : \mathbf{G}'' \quad \varepsilon : \mathbf{G}'' \sim \mathbf{G}}{\Delta; \Gamma \vdash \varepsilon (s[v/x]) :: \mathbf{G} : \mathbf{G}}$$

Thus, we are required to prove that $\Delta; \Gamma, x : \mathbf{G}' \vdash (\varepsilon s :: \mathbf{G}) : \mathbf{G}$. Or what is the same, by ($\mathbf{G}\text{asc}$) rule, we are required to prove that $\Delta; \Gamma, x : \mathbf{G}' \vdash s : \mathbf{G}''$. By the induction hypothesis on s , with $\Delta; \Gamma \vdash s[v/x] : \mathbf{G}''$, the result follows immediately.

Case (app). We know that $t = t_1 t_2$, and $(t_1 t_2)[v/x] = (t_1[v/x]) (t_2[v/x])$. Thus, we know that

$$(\mathbf{G}\text{app}) \frac{\Delta; \Gamma \vdash t_1[v/x] : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \quad \Delta; \Gamma \vdash t_2[v/x] : \mathbf{G}_1}{\Delta; \Gamma \vdash (t_1[v/x]) (t_2[v/x]) : \mathbf{G}_2}$$

where $\mathbf{G} = \mathbf{G}_2$. Therefore, we are required to prove that $\Delta; \Gamma, x : \mathbf{G}' \vdash t_1 t_2 : \mathbf{G}_2$. Or what is the same, by rule ($\mathbf{G}\text{app}$), we are required to prove that $\Delta; \Gamma, x : \mathbf{G}' \vdash t_1 : \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\Delta; \Gamma, x : \mathbf{G}' \vdash t_2 : \mathbf{G}_1$, which follow immediately by the induction hypothesis on t_1 and t_2 , with $\Delta; \Gamma \vdash t_1[v/x] : \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\Delta; \Gamma \vdash t_2[v/x] : \mathbf{G}_1$.

Case (pairi). We know that $t = \pi_i(t')$, and $(\pi_i(t'))[v/x] = \pi_i(t'[v/x])$. Thus, we know that

$$(\mathbf{G}\text{pairi}) \frac{\Delta; \Gamma \vdash t'[v/x] : \mathbf{G}_1 \times \mathbf{G}_2}{\Delta; \Gamma \vdash \pi_i(t'[v/x]) : \mathbf{G}_i}$$

where $\mathbf{G} = \mathbf{G}_i$. Therefore, we are required to prove that $\Delta; \Gamma, x : \mathbf{G}' \vdash \pi_i(t') : \mathbf{G}_i$. Or what is the same, by rule ($\mathbf{G}\text{pairi}$), we are required to prove that $\Delta; \Gamma, x : \mathbf{G}' \vdash t' : \mathbf{G}_1 \times \mathbf{G}_2$, which follow immediately by the induction hypothesis on t' , with $\Delta; \Gamma \vdash t'[v/x] : \mathbf{G}_1 \times \mathbf{G}_2$.

Case (appG). We know that $t = t' [\mathbf{F}]$, and $(t' [\mathbf{F}])[v/x] = (t'[v/x]) [\mathbf{F}]$. Thus, we know that

$$(\mathbf{G}\text{appG}) \frac{\Delta; \Gamma \vdash t'[v/x] : \forall X.\mathbf{G}'' \quad \Delta \vdash \mathbf{F}}{\Delta; \Gamma \vdash (t'[v/x]) [\mathbf{F}] : \mathbf{G}''[\mathbf{F}/\mathbf{X}]}$$

where $\mathbf{G} = \mathbf{G}''[\mathbf{F}/\mathbf{X}]$. Therefore, we are required to prove that $\Delta; \Gamma, x : \mathbf{G}' \vdash t' [\mathbf{F}] : \mathbf{G}''[\mathbf{F}/\mathbf{X}]$. Or what is the same, by the rule ($\mathbf{G}\text{appG}$), we are required to prove that $\Delta; \Gamma, x : \mathbf{G}' \vdash t' : \forall X.\mathbf{G}''$, which follows immediately by the induction hypotheses on t' , with $\Delta; \Gamma \vdash t'[v/x] : \forall X.\mathbf{G}''$.

□

Lemma B.57 $c : \mathbf{G} \sqsubseteq \mathbf{G}'$ if and only if $c[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}]$.

PROOF. We are required to prove that $c : \mathbf{G} \sqsubseteq \mathbf{G}'$ if and only if $c[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}]$. First, we prove that if $c : \mathbf{G} \sqsubseteq \mathbf{G}'$ then $c[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}]$. We proceed by induction on

Case ($\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$). We know that $\mathbf{c} = \mathbf{B}$, $\mathbf{G} = \mathbf{B}$ and $\mathbf{G}' = \mathbf{B}$. We know that

$$\frac{}{\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}}$$

Also, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}$, $\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}$. Thus, we are required to prove that $\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$, which follows immediately.

Case ($\mathbf{Y} : \mathbf{Y} \sqsubseteq \mathbf{Y}$). We know that $\mathbf{c} = \mathbf{Y}$, $\mathbf{G} = \mathbf{Y}$ and $\mathbf{G}' = \mathbf{Y}$. We know that

$$\frac{}{\mathbf{Y} : \mathbf{Y} \sqsubseteq \mathbf{Y}}$$

We have the following cases.

- $\mathbf{Y} = \mathbf{X}$. Therefore, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{X}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{F}$, $\mathbf{G}_{\mathbf{X} \mapsto \mathbf{F}} = \mathbf{X}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{F}$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{X}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{F}$. Thus, we are required to prove that $\mathbf{F} : \mathbf{F} \sqsubseteq \mathbf{F}$, which follows immediately.
- $\mathbf{Y} \neq \mathbf{X}$. Therefore, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}$, $\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}$. Thus, we are required to prove that $\mathbf{Y} : \mathbf{Y} \sqsubseteq \mathbf{Y}$, which follows immediately.

Case ($\mathbf{c} \rightarrow \mathbf{c}' : \mathbf{G}_1 \rightarrow \mathbf{G}'_1 \sqsubseteq \mathbf{G}_2 \rightarrow \mathbf{G}'_2$). We know that $\mathbf{c} = \mathbf{c}' \rightarrow \mathbf{c}''$, $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}'_1$ and $\mathbf{G}' = \mathbf{G}_2 \rightarrow \mathbf{G}'_2$. We know that

$$\frac{\mathbf{c}' : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2 \quad \mathbf{c}'' : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2}{\mathbf{c}' \rightarrow \mathbf{c}'' : \mathbf{G}_1 \rightarrow \mathbf{G}'_1 \sqsubseteq \mathbf{G}_2 \rightarrow \mathbf{G}'_2}$$

Also, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{c}' \rightarrow \mathbf{c}'')[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] \rightarrow \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}]$, $\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{G}_1 \rightarrow \mathbf{G}'_1)[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}] \rightarrow \mathbf{G}'_1[\mathbf{X} \mapsto \mathbf{F}]$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{G}_2 \rightarrow \mathbf{G}'_2)[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}] \rightarrow \mathbf{G}'_2[\mathbf{X} \mapsto \mathbf{F}]$. Thus, we are required to prove that $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}]$ and $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}'_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}'_2[\mathbf{X} \mapsto \mathbf{F}]$, which follows immediately by the induction hypothesis on $\mathbf{c}' : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$ and $\mathbf{c}'' : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$.

Case ($\mathbf{c}' \times \mathbf{c}'' : \mathbf{G}_1 \times \mathbf{G}'_1 \sqsubseteq \mathbf{G}_2 \times \mathbf{G}'_2$). We know that $\mathbf{c} = \mathbf{c}' \times \mathbf{c}''$, $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}'_1$ and $\mathbf{G}' = \mathbf{G}_2 \times \mathbf{G}'_2$. We know that

$$\frac{\mathbf{c}' : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2 \quad \mathbf{c}'' : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2}{\mathbf{c}' \times \mathbf{c}'' : \mathbf{G}_1 \times \mathbf{G}'_1 \sqsubseteq \mathbf{G}_2 \times \mathbf{G}'_2}$$

Also, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{c}' \times \mathbf{c}'')[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] \times \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}]$, $\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{G}_1 \times \mathbf{G}'_1)[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}] \times \mathbf{G}'_1[\mathbf{X} \mapsto \mathbf{F}]$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{G}_2 \times \mathbf{G}'_2)[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}] \times \mathbf{G}'_2[\mathbf{X} \mapsto \mathbf{F}]$. Thus, we are required to prove that $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}]$ and $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}'_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}'_2[\mathbf{X} \mapsto \mathbf{F}]$, which follows immediately by the induction hypothesis on $\mathbf{c}' : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$ and $\mathbf{c}'' : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$.

Case ($\forall \mathbf{Y}. \mathbf{c} : \forall \mathbf{Y}. \mathbf{G}_1 \sqsubseteq \forall \mathbf{Y}. \mathbf{G}_2$). We know that $\mathbf{c} = \forall \mathbf{Y}. \mathbf{c}'$, $\mathbf{G} = \forall \mathbf{Y}. \mathbf{G}_1$ and $\mathbf{G}' = \forall \mathbf{Y}. \mathbf{G}_2$. We know that

$$\frac{\mathbf{c}' : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2}{\forall \mathbf{Y}. \mathbf{c} : \forall \mathbf{Y}. \mathbf{G}_1 \sqsubseteq \forall \mathbf{Y}. \mathbf{G}_2}$$

Also, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = (\forall \mathbf{Y}. \mathbf{c}')[\mathbf{X} \mapsto \mathbf{F}] = \forall \mathbf{Y}. \mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]$, $\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] = (\forall \mathbf{Y}. \mathbf{G}_1)[\mathbf{X} \mapsto \mathbf{F}] = \forall \mathbf{Y}. \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}]$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = (\forall \mathbf{Y}. \mathbf{G}_2)[\mathbf{X} \mapsto \mathbf{F}] = \forall \mathbf{Y}. \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}]$. Thus, we are required to prove that $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}]$, which follows immediately by the induction hypothesis on $\mathbf{c}' : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$.

Case ($\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta}$). We know that $\mathbf{c} = \text{inj}_{\mathbf{B}}$, $\mathbf{G} = \mathbf{B}$ and $\mathbf{G}' = ?_{\delta}$. We know that

$$\frac{}{\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta}}$$

Also, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_B[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_B$, $\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = ?_\delta[\mathbf{X} \mapsto \mathbf{F}] = ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$. Thus, we are required to prove that $\mathbf{inj}_B : \mathbf{B} \sqsubseteq ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$, which follows immediately.

Case $(\mathbf{inj}_Y : \mathbf{F}' \sqsubseteq ?_\delta)$. We know that $\mathbf{c} = \mathbf{inj}_Y$, $\mathbf{G} = \mathbf{F}'$ and $\mathbf{G}' = ?_\delta$. We know that

$$\frac{\mathbf{Y} : \mathbf{F}' \in \delta}{\mathbf{inj}_Y : \mathbf{F}' \sqsubseteq ?_\delta}$$

We have the following cases.

- $\mathbf{F}' = \mathbf{X}$. Thus, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_Y[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_Y$, $\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{X}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{F}$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = ?_\delta[\mathbf{X} \mapsto \mathbf{F}] = ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$. Thus, we are required to prove that $\mathbf{inj}_Y : \mathbf{F} \sqsubseteq ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$, which follows immediately because $\mathbf{Y} : (\mathbf{X}[\mathbf{X} \mapsto \mathbf{F}]) = \mathbf{Y} : \mathbf{F} \in ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$.
- $\mathbf{F}' \neq \mathbf{X}$. Thus, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_Y[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_Y$, $\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{F}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{F}'$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = ?_\delta[\mathbf{X} \mapsto \mathbf{F}] = ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$. Thus, we are required to prove that $\mathbf{inj}_Y : \mathbf{F}' \sqsubseteq ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$, which follows immediately because $\mathbf{Y} : (\mathbf{F}'[\mathbf{X} \mapsto \mathbf{F}]) = \mathbf{Y} : \mathbf{F}' \in ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$.

Case $(\mathbf{inj}_\rightarrow(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_\delta)$. We know that $\mathbf{c} = \mathbf{inj}_\rightarrow(\mathbf{c}')$ and $\mathbf{G}' = ?_\delta$. We know that

$$\frac{\mathbf{c}' : \mathbf{G} \sqsubseteq ?_\delta \rightarrow ?_\delta}{\mathbf{inj}_\rightarrow(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_\delta}$$

Also, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_\rightarrow(\mathbf{c}')[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_\rightarrow(\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}])$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = ?_\delta[\mathbf{X} \mapsto \mathbf{F}] = ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$. Thus, we are required to prove that $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]} \rightarrow ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$, which follows immediately by the induction hypothesis on $\mathbf{c}' : \mathbf{G} \sqsubseteq ?_\delta \rightarrow ?_\delta$.

Case $(\mathbf{inj}_\times(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_\delta)$. We know that $\mathbf{c} = \mathbf{inj}_\times(\mathbf{c}')$ and $\mathbf{G}' = ?_\delta$. We know that

$$\frac{\mathbf{c}' : \mathbf{G} \sqsubseteq ?_\delta \times ?_\delta}{\mathbf{inj}_\times(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_\delta}$$

Also, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_\times(\mathbf{c}')[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_\times(\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}])$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = ?_\delta[\mathbf{X} \mapsto \mathbf{F}] = ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$. Thus, we are required to prove that $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]} \times ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$, which follows immediately by the induction hypothesis on $\mathbf{c}' : \mathbf{G} \sqsubseteq ?_\delta \times ?_\delta$.

Case $(\mathbf{inj}_\forall(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_\delta)$. We know that $\mathbf{c} = \mathbf{inj}_\forall(\mathbf{c}')$ and $\mathbf{G}' = ?_\delta$. We know that

$$\frac{\mathbf{c}' : \mathbf{G} \sqsubseteq \forall \mathbf{Y}. ?_{\delta, \mathbf{Y} : \mathbf{Y}}}{\mathbf{inj}_\forall(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_\delta}$$

Also, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_\forall(\mathbf{c}')[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_\forall(\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}])$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = ?_\delta[\mathbf{X} \mapsto \mathbf{F}] = ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$. Thus, we are required to prove that $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \forall \mathbf{Y}. ?_{(\delta, \mathbf{Y} : \mathbf{Y})[\mathbf{X} \mapsto \mathbf{F}]}$, which follows immediately by the induction hypothesis on $\mathbf{c}' : \mathbf{G} \sqsubseteq \forall \mathbf{Y}. ?_{\delta, \mathbf{Y} : \mathbf{Y}}$. Note that $?_{(\delta, \mathbf{Y} : \mathbf{Y})[\mathbf{X} \mapsto \mathbf{F}]} = ?_{(\delta[\mathbf{X} \mapsto \mathbf{F}]), \mathbf{Y} : \mathbf{Y}}$.

Case $(\mathbf{inj}_? : ?_\delta \sqsubseteq ?_{\delta'})$. We know that $\mathbf{c} = \mathbf{inj}_?$, $\mathbf{G} = ?_\delta$. $\mathbf{G}' = ?_{\delta'}$. We know that

$$\frac{\delta \sqsubseteq \delta'}{\mathbf{inj}_? : ?_\delta \sqsubseteq ?_{\delta'}}$$

Also, we know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_?[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{inj}_?$, $\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] = ?_\delta[\mathbf{X} \mapsto \mathbf{F}] = ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]}$ and $\mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] = ?_{\delta'}[\mathbf{X} \mapsto \mathbf{F}] = ?_{\delta'[\mathbf{X} \mapsto \mathbf{F}]}$. Thus, we are required to prove that $\mathbf{inj}_? : ?_{\delta[\mathbf{X} \mapsto \mathbf{F}]} \sqsubseteq ?_{\delta'[\mathbf{X} \mapsto \mathbf{F}]}$, which follows immediately since $\delta \sqsubseteq \delta'$ then $\delta[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \delta'[\mathbf{X} \mapsto \mathbf{F}]$.

Next, we prove that if $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}]$ then $\mathbf{c} : \mathbf{G} \sqsubseteq \mathbf{G}'$. We proceed by induction on \mathbf{c} . The proof is very similar to the first one. \square

Lemma B.58 *If $\Delta \vdash \mathbf{F}$, then $\varepsilon : \mathbf{G}' \sim \mathbf{G}$ if and only if $\varepsilon[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}'[\mathbf{F}/\mathbf{X}] \sim \mathbf{G}[\mathbf{F}/\mathbf{X}]$.*

PROOF. We proceed by proving that for all $(\mathbf{G}^*, \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon$, we get that $[(\mathbf{G}^*, \mathbf{c}_1, \mathbf{c}_2)] : \mathbf{G}' \sim \mathbf{G}$ if and only if $[(\mathbf{G}^*, \mathbf{c}_1, \mathbf{c}_2)][\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}] \sim \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}]$. Or what is the same, we are required to prove that $\mathbf{c}_1 : \mathbf{G}^* \sqsubseteq \mathbf{G}'$ if and only if $\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}^*[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}'[\mathbf{X} \mapsto \mathbf{F}]$ and $\mathbf{c}_2 : \mathbf{G}^* \sqsubseteq \mathbf{G}$ if and only if $\mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}^*[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}[\mathbf{X} \mapsto \mathbf{F}]$. By Lemma B.57, the result follows immediately. \square

Lemma B.59 *If $\varepsilon : \mathbf{G}' \sim \mathbf{G}$ and $(n, \rho) \in \mathcal{D}[\Delta]$, then $\rho_1(\varepsilon) : \rho_1(\mathbf{G}') \sim \rho_1(\mathbf{G})$ and $\rho_2(\varepsilon) : \rho_2(\mathbf{G}') \sim \rho_2(\mathbf{G})$.*

PROOF. We know that $\varepsilon : \mathbf{G}' \sim \mathbf{G}$ and $(n, \rho) \in \mathcal{D}[\Delta]$. We are required to prove that $\rho_1(\varepsilon) : \rho_1(\mathbf{G}') \sim \rho_1(\mathbf{G})$ and $\rho_2(\varepsilon) : \rho_2(\mathbf{G}') \sim \rho_2(\mathbf{G})$. We proceed by induction on Δ .

- $\Delta = \emptyset$. By the definition of $(n, \rho) \in \mathcal{D}[\Delta]$, we know that $\rho = \emptyset$. Thus, we know that $\rho_i(\varepsilon) = \varepsilon$, $\rho_i(\mathbf{G}') = \mathbf{G}'$ and $\rho_i(\mathbf{G}) = \mathbf{G}$. Therefore, we are required to prove that $\varepsilon : \mathbf{G}' \sim \mathbf{G}$, which follows immediately by the premise.
- $\Delta = \Delta', \mathbf{X}$. By the definition of $(n, \rho) \in \mathcal{D}[\Delta]$, we know that $\rho = \rho'[X \mapsto (\mathbf{B}_1, \mathbf{B}_2, R)]$ and $(n, \rho') \in \mathcal{D}[\Delta']$. We know that $\rho_i(\varepsilon) = \rho'_i(\varepsilon[\mathbf{X} \mapsto \mathbf{B}_i])$, $\rho_i(\mathbf{G}') = \rho'_i(\mathbf{G}'[\mathbf{B}_i/\mathbf{X}])$ and $\rho_i(\mathbf{G}) = \rho'_i(\mathbf{G}[\mathbf{B}_i/\mathbf{X}])$. Thus, we are required to prove that $\rho'_i(\varepsilon[\mathbf{X} \mapsto \mathbf{B}_i]) : \rho'_i(\mathbf{G}'[\mathbf{B}_i/\mathbf{X}]) \sim \rho'_i(\mathbf{G}[\mathbf{B}_i/\mathbf{X}])$. By Lemma B.58, we know that $\varepsilon[\mathbf{X} \mapsto \mathbf{B}_1] : \mathbf{G}'[\mathbf{B}_1/\mathbf{X}] \sim \mathbf{G}[\mathbf{B}_1/\mathbf{X}]$ and $\varepsilon[\mathbf{X} \mapsto \mathbf{B}_2] : \mathbf{G}'[\mathbf{B}_2/\mathbf{X}] \sim \mathbf{G}[\mathbf{B}_2/\mathbf{X}]$. By the induction hypothesis on Δ' , with $(n, \rho') \in \mathcal{D}[\Delta']$ and evidence $\varepsilon[\mathbf{X} \mapsto \mathbf{B}_1] : \mathbf{G}'[\mathbf{B}_1/\mathbf{X}] \sim \mathbf{G}[\mathbf{B}_1/\mathbf{X}]$ and $\varepsilon[\mathbf{X} \mapsto \mathbf{B}_2] : \mathbf{G}'[\mathbf{B}_2/\mathbf{X}] \sim \mathbf{G}[\mathbf{B}_2/\mathbf{X}]$, we get that $\rho'_1(\varepsilon[\mathbf{X} \mapsto \mathbf{B}_1]) : \rho'_1(\mathbf{G}'[\mathbf{B}_1/\mathbf{X}]) \sim \rho'_1(\mathbf{G}[\mathbf{B}_1/\mathbf{X}])$ and $\rho'_2(\varepsilon[\mathbf{X} \mapsto \mathbf{B}_2]) : \rho'_2(\mathbf{G}'[\mathbf{B}_2/\mathbf{X}]) \sim \rho'_2(\mathbf{G}[\mathbf{B}_2/\mathbf{X}])$. Thus, the result follows immediately. \square

Lemma B.60 *If $\Delta; \Gamma \vdash \mathbf{t} : \mathbf{G}$, $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$, then $\vdash \rho_1(\gamma_1(\mathbf{t})) : \rho_1(\mathbf{G})$ and $\vdash \rho_2(\gamma_2(\mathbf{t})) : \rho_2(\mathbf{G})$.*

PROOF. We know that $\Delta; \Gamma \vdash \mathbf{t} : \mathbf{G}$, $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma) \in \mathcal{G}_\rho[\Gamma]$. We are required to prove that $\vdash \rho_1(\gamma_1(\mathbf{t})) : \rho_1(\mathbf{G})$ and $\vdash \rho_2(\gamma_2(\mathbf{t})) : \rho_2(\mathbf{G})$. We proceed by induction on $\Delta; \Gamma \vdash \mathbf{t} : \mathbf{G}$.

Case (Gb). We know that $\mathbf{t} = \mathbf{b}$, $\mathbf{G} = \mathbf{B}$, and

$$(Gb) \frac{\theta(\mathbf{b}) = \mathbf{B} \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \mathbf{b} : \mathbf{B}}$$

Also, we know that $\rho_i(\gamma_i(\mathbf{t})) = \rho_i(\gamma_i(\mathbf{b})) = \mathbf{b}$ and $\rho_i(\mathbf{G}) = \rho_i(\mathbf{B}) = \mathbf{B}$. Therefore, we are required to prove that $\vdash \mathbf{b} : \mathbf{B}$, which follows immediately.

Case (\mathbf{G}_x). We know that $\mathbf{t} = x$, and

$$(\mathbf{G}_x) \frac{x : \mathbf{G} \in \mathbf{\Gamma} \quad \Delta \vdash \mathbf{\Gamma}}{\Delta; \mathbf{\Gamma} \vdash x : \mathbf{G}}$$

Since $x : \mathbf{G} \in \mathbf{\Gamma}$ and $(n, \gamma) \in \mathcal{G}_\rho[\mathbf{\Gamma}]$, we know that $(x \mapsto (\mathbf{v}_1, \mathbf{v}_2)) \in \gamma$, and $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}]$. Therefore, we know that $\vdash \mathbf{v}_i : \rho_i(\mathbf{G})$ and $\rho_i(\gamma_i(\mathbf{t})) = \rho_i(\gamma_i(x)) = \mathbf{v}_i$. Thus, we are required to prove that $\vdash \mathbf{v}_i : \rho_i(\mathbf{G})$, which follows immediately.

Case (\mathbf{G}_λ). We know that $\mathbf{t} = \lambda x : \mathbf{G}_1.t'$, $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$, and

$$(\mathbf{G}_\lambda) \frac{\Delta; \mathbf{\Gamma}, x : \mathbf{G}_1 \vdash t' : \mathbf{G}_2}{\Delta; \mathbf{\Gamma} \vdash (\lambda x : \mathbf{G}_1.t') : \mathbf{G}_1 \rightarrow \mathbf{G}_2}$$

We know that $\rho_i(\gamma_i(\lambda x : \mathbf{G}_1.t')) = \lambda x : \rho_i(\mathbf{G}_1). \rho_i(\gamma_i(t'))$ and $\rho_i(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \rho_i(\mathbf{G}_1) \rightarrow \rho_i(\mathbf{G}_2)$. We are required to prove that $\vdash \lambda x : \rho_i(\mathbf{G}_1). \rho_i(\gamma_i(t')) : \rho_i(\mathbf{G}_1) \rightarrow \rho_i(\mathbf{G}_2)$. Or what is the same, by rule (\mathbf{G}_λ), we are required to prove $x : \rho_i(\mathbf{G}_1) \vdash \rho_i(\gamma_i(t')) : \rho_i(\mathbf{G}_2)$. By the induction hypothesis on $\Delta; \mathbf{\Gamma}, x : \mathbf{G}_1 \vdash t' : \mathbf{G}_2$, with $(n, \rho) \in \mathcal{D}[\Delta]$ and $(n, \gamma[(x \mapsto (\mathbf{v}_1, \mathbf{v}_2))]) \in \mathcal{G}_\rho[\mathbf{\Gamma}, x : \mathbf{G}_1]$, where $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}_1]$, we know that $\vdash \rho_i(\gamma_i(t'))[\mathbf{v}_i/x] : \rho_i(\mathbf{G}_2)$. Since $(n, \mathbf{v}_1, \mathbf{v}_2) \in \mathcal{V}_\rho[\mathbf{G}_1]$, we know that $\vdash \mathbf{v}_i : \rho_i(\mathbf{G}_1)$. By Lemma B.56, we get that $x : \rho_i(\mathbf{G}_1) \vdash \rho_i(\gamma_i(t')) : \rho_i(\mathbf{G}_2)$, and the result follows immediately.

Case (\mathbf{G}_{pair}). We know that $\mathbf{t} = \langle \mathbf{s}_1, \mathbf{s}_2 \rangle$, $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$, and

$$(\mathbf{G}_{\text{pair}}) \frac{\Delta; \mathbf{\Gamma} \vdash \mathbf{s}_1 : \mathbf{G}_1 \quad \Delta; \mathbf{\Gamma} \vdash \mathbf{s}_2 : \mathbf{G}_2}{\Delta; \mathbf{\Gamma} \vdash \langle \mathbf{s}_1, \mathbf{s}_2 \rangle : \mathbf{G}_1 \times \mathbf{G}_2}$$

We know that $\rho_i(\gamma_i(\langle \mathbf{s}_1, \mathbf{s}_2 \rangle)) = \langle \rho_i(\gamma_i(\mathbf{s}_1)), \rho_i(\gamma_i(\mathbf{s}_2)) \rangle$ and $\rho_i(\mathbf{G}_1 \times \mathbf{G}_2) = \rho_i(\mathbf{G}_1) \times \rho_i(\mathbf{G}_2)$. We are required to prove that $\vdash \langle \rho_i(\gamma_i(\mathbf{s}_1)), \rho_i(\gamma_i(\mathbf{s}_2)) \rangle : \rho_i(\mathbf{G}_1) \times \rho_i(\mathbf{G}_2)$. Or what is the same, by rule (\mathbf{G}_{pair}), we are required to prove that $\vdash \rho_i(\gamma_i(\mathbf{s}_1)) : \rho_i(\mathbf{G}_1)$ and $\vdash \rho_i(\gamma_i(\mathbf{s}_2)) : \rho_i(\mathbf{G}_2)$. By the induction hypothesis on $\Delta; \mathbf{\Gamma} \vdash \mathbf{s}_1 : \mathbf{G}_1$ and $\Delta; \mathbf{\Gamma} \vdash \mathbf{s}_2 : \mathbf{G}_2$, the result follows immediately.

Case (\mathbf{G}_Λ). We know that $\mathbf{t} = \Lambda X.t'$, $\mathbf{G} = \forall X.\mathbf{G}'$, and

$$(\mathbf{G}_\Lambda) \frac{\Delta, X; \mathbf{\Gamma} \vdash t' : \mathbf{G}' \quad \Delta \vdash \mathbf{\Gamma}}{\Delta; \mathbf{\Gamma} \vdash (\Lambda X.t') : \forall X.\mathbf{G}'}$$

We know that $\rho_i(\gamma_i(\Lambda X.t')) = \Lambda X. \rho_i(\gamma_i(t'))$ and $\rho_i(\forall X.\mathbf{G}') = \forall X. \rho_i(\mathbf{G}')$. We are required to prove that $\vdash \Lambda X. \rho_i(\gamma_i(t')) : \forall X. \rho_i(\mathbf{G}')$. Or what is the same, by rule (\mathbf{G}_Λ), we are required to prove that $X \vdash \rho_i(\gamma_i(t')) : \rho_i(\mathbf{G}')$. By the induction hypothesis on $\Delta, X; \mathbf{\Gamma} \vdash t' : \mathbf{G}'$, with $(n, \rho, [(X \mapsto (\mathbf{B}, \mathbf{B}, \mathcal{V}_\emptyset[\mathbf{B}]])]) \in \mathcal{D}[\Delta, X]$ and $(n, \gamma) \in \mathcal{G}_\rho[\mathbf{\Gamma}]$, we get that $\vdash \rho_i(\gamma_i(t'))[X \mapsto \mathbf{B}] : \rho_i(\mathbf{G}')[X \mapsto \mathbf{B}]$. Thus, by Lemma B.55, we get that $X \vdash \rho_i(\gamma_i(t')) : \rho_i(\mathbf{G}')$, and the result follows immediately.

Case (\mathbf{G}_{asc}). We know that $\mathbf{t} = \varepsilon s :: \mathbf{G}$, and

$$(\mathbf{G}_{\text{asc}}) \frac{\Delta; \mathbf{\Gamma} \vdash s : \mathbf{G}' \quad \varepsilon : \mathbf{G}' \sim \mathbf{G}}{\Delta; \mathbf{\Gamma} \vdash \varepsilon s :: \mathbf{G} : \mathbf{G}}$$

We know that $\rho_i(\gamma_i(\varepsilon s :: \mathbf{G})) = \rho_i(\varepsilon) \rho_i(\gamma_i(s)) :: \rho_i(\mathbf{G})$. Thus, we are required to prove that $\vdash \rho_i(\varepsilon) \rho_i(\gamma_i(s)) :: \rho_i(\mathbf{G}) : \rho_i(\mathbf{G})$. By (\mathbf{G}_{asc}), we are required to prove that

- $\rho_1(\varepsilon) : \rho_1(\mathbf{G}') \sim \rho_1(\mathbf{G})$ and $\rho_2(\varepsilon) : \rho_2(\mathbf{G}') \sim \rho_2(\mathbf{G})$, which follow immediately by Lemma B.59.
- $\vdash \rho_i(\gamma_i(s)) : \rho_i(\mathbf{G}')$, which follows by the induction hypothesis on $\Delta; \mathbf{\Gamma} \vdash s : \mathbf{G}'$.

Case (\mathbf{G}_{app}). We know that $\mathbf{t} = \mathbf{t}_1 \mathbf{t}_2$ and $\mathbf{G} = \mathbf{G}_2$, and

$$(\mathbf{G}_{\text{app}}) \frac{\Delta; \mathbf{\Gamma} \vdash \mathbf{t}_1 : \mathbf{G}_1 \rightarrow \mathbf{G}_2 \quad \Delta; \mathbf{\Gamma} \vdash \mathbf{t}_2 : \mathbf{G}_1}{\Delta; \mathbf{\Gamma} \vdash \mathbf{t}_1 \mathbf{t}_2 : \mathbf{G}_2}$$

We know that $\rho_i(\gamma_i(\mathbf{t}_1 \ \mathbf{t}_2)) = \rho_i(\gamma_i(\mathbf{t}_1)) \ \rho_i(\gamma_i(\mathbf{t}_2))$ and $\rho_i(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \rho_i(\mathbf{G}_1) \rightarrow \rho_i(\mathbf{G}_2)$. Therefore, we are required to prove that $\vdash \rho_i(\gamma_i(\mathbf{t}_1)) \ \rho_i(\gamma_i(\mathbf{t}_2)) : \rho_i(\mathbf{G}_2)$. Or what is the same, by rule (**Gapp**), we are required to prove that $\vdash \rho_i(\gamma_i(\mathbf{t}_1)) : \rho_i(\mathbf{G}_1) \rightarrow \rho_i(\mathbf{G}_2)$ and $\vdash \rho_i(\gamma_i(\mathbf{t}_2)) : \rho_i(\mathbf{G}_1)$, which follow immediately by the induction hypothesis on $\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\Delta; \Gamma \vdash \mathbf{t}_2 : \mathbf{G}_1$. *Case* (**Gpairi**). We know that $\mathbf{t} = \pi_i(\mathbf{t}')$ and $\mathbf{G} = \mathbf{G}_i$, and

$$(\mathbf{Gpair}i) \frac{\Delta; \Gamma \vdash \mathbf{t}' : \mathbf{G}_1 \times \mathbf{G}_2}{\Delta; \Gamma \vdash \pi_i(\mathbf{t}') : \mathbf{G}_i}$$

We know that $\rho_1(\gamma_1(\pi_i(\mathbf{t}'))) = \pi_i(\rho_1(\gamma_1(\mathbf{t}')))$, $\rho_2(\gamma_2(\pi_i(\mathbf{t}'))) = \pi_i(\rho_2(\gamma_2(\mathbf{t}')))$, $\rho_1(\mathbf{G}_1 \times \mathbf{G}_2) = \rho_1(\mathbf{G}_1) \times \rho_1(\mathbf{G}_2)$ and $\rho_2(\mathbf{G}_1 \times \mathbf{G}_2) = \rho_2(\mathbf{G}_1) \times \rho_2(\mathbf{G}_2)$. Therefore, we are required to prove that $\vdash \pi_i(\rho_1(\gamma_1(\mathbf{t}'))) : \rho_1(\mathbf{G}_i)$ and $\vdash \pi_i(\rho_2(\gamma_2(\mathbf{t}'))) : \rho_2(\mathbf{G}_i)$. Or what is the same, by rule (**Gpairi**), we are required to prove that $\vdash \rho_1(\gamma_1(\mathbf{t}')) : \rho_1(\mathbf{G}_1) \times \rho_1(\mathbf{G}_2)$ and $\vdash \rho_2(\gamma_2(\mathbf{t}')) : \rho_2(\mathbf{G}_1) \times \rho_2(\mathbf{G}_2)$, which follow immediately by the induction hypothesis on $\Delta; \Gamma \vdash \mathbf{t}' : \mathbf{G}_1 \times \mathbf{G}_2$.

Case (**GappG**). We know that $\mathbf{t} = \mathbf{t}'[\mathbf{F}]$ and $\mathbf{G} = \mathbf{G}'[\mathbf{F}/\mathbf{X}]$, and

$$(\mathbf{Gapp}G) \frac{\Delta; \Gamma \vdash \mathbf{t}' : \forall \mathbf{X}. \mathbf{G}' \quad \Delta \vdash \mathbf{F}}{\Delta; \Gamma \vdash \mathbf{t}'[\mathbf{F}] : \mathbf{G}'[\mathbf{F}/\mathbf{X}]}$$

We know that $\rho_i(\gamma_i(\mathbf{t}'[\mathbf{F}])) = \rho_i(\gamma_i(\mathbf{t}'))[\rho_i(\mathbf{F})]$ and $\rho_i(\mathbf{G}'[\mathbf{F}/\mathbf{X}]) = \rho_i(\mathbf{G}')[\rho_i(\mathbf{F})/\mathbf{X}]$. Therefore, we are required to prove that $\vdash \rho_i(\gamma_i(\mathbf{t}'))[\rho_i(\mathbf{F})] : \rho_i(\mathbf{G}')[\rho_i(\mathbf{F})/\mathbf{X}]$. Or what is the same, by the rule (**GappG**), we are required to prove that $\vdash \rho_i(\gamma_i(\mathbf{t}')) : \forall \mathbf{X}. \rho_i(\mathbf{G}')$, which follows immediately by the induction hypotheses on $\Delta; \Gamma \vdash \mathbf{t}' : \forall \mathbf{X}. \mathbf{G}'$.

□

Lemma B.61 *If $\varepsilon : \mathbf{G} \sim ?_\delta$, then there exists \mathbf{G}_R and ε' such that $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$ and $\varepsilon \dot{\;} \varepsilon'$ is defined.*

PROOF. We know that $\varepsilon : \mathbf{G} \sim ?_\delta$. We are required to prove that there exists \mathbf{G}_R and ε' such that $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$ and $\varepsilon \dot{\;} \varepsilon'$ is defined. We proceed by the shape of the evidence.

- $(\mathbf{B}, \mathbf{B}, \mathbf{B}) \in \varepsilon$. This case is not possible because $\{(\mathbf{B}, \mathbf{B}, \mathbf{B})\} : \mathbf{B} \sim \mathbf{B}$. It is not the case that $\{(\mathbf{B}, \mathbf{B}, \mathbf{B})\} : \mathbf{B} \sim ?_\delta$.
- $(\mathbf{B}, \mathbf{B}, \text{inj}_\mathbf{B}) \in \varepsilon$. Then, we can take $\mathbf{G}_R = \mathbf{B}$. Note that
 - $\varepsilon' = (\mathbf{B}, \text{inj}_\mathbf{B}, \mathbf{B})$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$.
 - $\varepsilon'' = \varepsilon \dot{\;} \varepsilon'$ is defined because $[(\mathbf{B}, \text{inj}_\mathbf{B}, \mathbf{B})] \dot{\;} [(\mathbf{B}, \text{inj}_\mathbf{B}, \mathbf{B})] = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\} \in \varepsilon''$. Thus, the result follows.
- $(\mathbf{B}, \text{inj}_\mathbf{B}, \text{inj}_\mathbf{B}) \in \varepsilon$. Then, we can take $\mathbf{G}_R = \mathbf{B}$. Note that
 - $\varepsilon' = (\mathbf{B}, \text{inj}_\mathbf{B}, \mathbf{B})$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$.
 - $\varepsilon'' = \varepsilon \dot{\;} \varepsilon'$ is defined because $[(\mathbf{B}, \text{inj}_\mathbf{B}, \text{inj}_\mathbf{B})] \dot{\;} [(\mathbf{B}, \text{inj}_\mathbf{B}, \mathbf{B})] = [(\mathbf{B}, \text{inj}_\mathbf{B}, \mathbf{B})] \in \varepsilon''$. Thus, the result follows.
- $(\mathbf{B}, \mathbf{B}, \text{inj}_\mathbf{X}) \in \varepsilon$. Then, we know that $\mathbf{X} : \mathbf{B} \in \delta$ and we can take $\mathbf{G}_R = \mathbf{X}$. Note that
 - $\varepsilon' = (\mathbf{B}, \text{inj}_\mathbf{X}, \mathbf{B})$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$.

- $\varepsilon'' = \varepsilon \circledast \varepsilon'$ is defined because $[(\mathbf{B}, \mathbf{B}, \text{inj}_X)] \circledast [(\mathbf{B}, \text{inj}_X, \mathbf{B})] = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\} \in \varepsilon''$. Thus, the result follows.
- $(\mathbf{X}, \mathbf{X}, \mathbf{X}) \in \varepsilon$. This case is not possible because $\{(\mathbf{X}, \mathbf{X}, \mathbf{X})\} : \mathbf{X} \sim \mathbf{X}$. It is not the case that $\{(\mathbf{X}, \mathbf{X}, \mathbf{X})\} : \mathbf{X} \sim ?_\delta$.
- $(\mathbf{X}, \mathbf{X}, \text{inj}_Y) \in \varepsilon$. Then, we know that $\mathbf{Y} : \mathbf{X} \in \delta$ and we can take $\mathbf{G}_R = \mathbf{Y}$. Note that
 - $\varepsilon' = (\mathbf{X}, \text{inj}_Y, \mathbf{X})$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$.
 - $\varepsilon'' = \varepsilon \circledast \varepsilon'$ is defined because $[(\mathbf{X}, \mathbf{X}, \text{inj}_Y)] \circledast [(\mathbf{X}, \text{inj}_Y, \mathbf{X})] = \{(\mathbf{X}, \mathbf{X}, \mathbf{X})\} \in \varepsilon''$. Thus, the result follows.
- $(\mathbf{F}, \text{inj}_Z, \text{inj}_Y) \in \varepsilon$. Then, we know that $\mathbf{Z} : \mathbf{F} \in \delta$, $\mathbf{Y} : \mathbf{F} \in \delta$ and we can take $\mathbf{G}_R = \mathbf{Y}$. Note that
 - $\varepsilon' = (\mathbf{F}, \text{inj}_Y, \mathbf{F})$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$.
 - $\varepsilon'' = \varepsilon \circledast \varepsilon'$ is defined because $[(\mathbf{F}, \text{inj}_Z, \text{inj}_Y)] \circledast [(\mathbf{F}, \text{inj}_Y, \mathbf{F})] = [(\mathbf{F}, \text{inj}_Z, \mathbf{F})] \in \varepsilon''$. Thus, the result follows.

- $(\mathbf{G}_1 \rightarrow \mathbf{G}_2, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22}) \in \varepsilon$. This case is not possible because

$$[(\mathbf{G}_1 \rightarrow \mathbf{G}_2, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22})] : \mathbf{G}_{11} \rightarrow \mathbf{G}_{12} \sim \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}$$

It is not the case that

$$[(\mathbf{G}_1 \rightarrow \mathbf{G}_2, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22})] : \mathbf{G}_{11} \rightarrow \mathbf{G}_{12} \sim ?_\delta$$

- $(\mathbf{G}_1 \times \mathbf{G}_2, \mathbf{c}_{11} \times \mathbf{c}_{12}, \mathbf{c}_{21} \times \mathbf{c}_{22}) \in \varepsilon$. This case is not possible because

$$[(\mathbf{G}_1 \times \mathbf{G}_2, \mathbf{c}_{11} \times \mathbf{c}_{12}, \mathbf{c}_{21} \times \mathbf{c}_{22})] : \mathbf{G}_{11} \times \mathbf{G}_{12} \sim \mathbf{G}_{21} \times \mathbf{G}_{22}$$

It is not the case that

$$[(\mathbf{G}_1 \times \mathbf{G}_2, \mathbf{c}_{11} \times \mathbf{c}_{12}, \mathbf{c}_{21} \times \mathbf{c}_{22})] : \mathbf{G}_{11} \times \mathbf{G}_{12} \sim ?_\delta$$

- $(\forall \mathbf{X}. \mathbf{G}_1, \forall \mathbf{X}. \mathbf{c}_{11}, \forall \mathbf{X}. \mathbf{c}_{21}) \in \varepsilon$. This case is not possible because

$$[(\forall \mathbf{X}. \mathbf{G}_1, \forall \mathbf{X}. \mathbf{c}_{11}, \forall \mathbf{X}. \mathbf{c}_{21})] : \forall \mathbf{X}. \mathbf{G}_{11} \sim \forall \mathbf{X}. \mathbf{G}_{21}$$

It is not the case that

$$[(\forall \mathbf{X}. \mathbf{G}_1, \forall \mathbf{X}. \mathbf{c}_{11}, \forall \mathbf{X}. \mathbf{c}_{21})] : \forall \mathbf{X}. \mathbf{G}_{11} \sim ?_\delta$$

- $(\mathbf{G}_1 \rightarrow \mathbf{G}_2, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \text{inj}_{\rightarrow}(\mathbf{c}_{21} \rightarrow \mathbf{c}_{22})) \in \varepsilon$. Then, we can take $\mathbf{G}_R = ?_\delta \rightarrow ?_\delta$. Note that

- $\varepsilon' = (?_\delta \rightarrow ?_\delta, \text{inj}_{\rightarrow}(\text{inj}_{\rightarrow} \rightarrow \text{inj}_{\rightarrow}), \text{inj}_{\rightarrow} \rightarrow \text{inj}_{\rightarrow})$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$.
- $\varepsilon'' = \varepsilon \circledast \varepsilon'$ is defined because

$$\begin{aligned} & [(\mathbf{G}_1 \rightarrow \mathbf{G}_2, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \text{inj}_{\rightarrow}(\mathbf{c}_{21} \rightarrow \mathbf{c}_{22}))] \circledast [(?_\delta \rightarrow ?_\delta, \text{inj}_{\rightarrow}(\text{inj}_{\rightarrow} \rightarrow \text{inj}_{\rightarrow}), \text{inj}_{\rightarrow} \rightarrow \text{inj}_{\rightarrow})] = \\ & [(\mathbf{G}_1 \rightarrow \mathbf{G}_2, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22})] \in \varepsilon'' \end{aligned}$$

by Lemma B.41. Thus, the result follows.

- $(\mathbf{G}_1 \times \mathbf{G}_2, \mathbf{c}_{11} \times \mathbf{c}_{12}, \text{inj}_{\times}(\mathbf{c}_{21} \times \mathbf{c}_{22})) \in \varepsilon$. Then, we can take $\mathbf{G}_R = ?_\delta \times ?_\delta$. Note that

- $\varepsilon' = (?_{\delta} \times ?_{\delta}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_{\mathbf{R}}$.
- $\varepsilon'' = \varepsilon \circledast \varepsilon'$ is defined because

$$\begin{aligned} & [(\mathbf{G}_1 \times \mathbf{G}_2, \mathbf{c}_{11} \times \mathbf{c}_{12}, \text{inj}_{\times}(\mathbf{c}_{21} \times \mathbf{c}_{22}))] \circledast [(?_{\delta} \times ?_{\delta}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})] = \\ & \quad [(\mathbf{G}_1 \times \mathbf{G}_2, \mathbf{c}_{11} \times \mathbf{c}_{12}, \mathbf{c}_{21} \times \mathbf{c}_{22})] \in \varepsilon'' \end{aligned}$$

by Lemma B.41. Thus, the result follows.

- $(\forall \mathbf{X}.\mathbf{G}_1, \forall \mathbf{X}.\mathbf{c}_{11}, \text{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_{21})) \in \varepsilon$. Then, we can take $\mathbf{G}_{\mathbf{R}} = \forall \mathbf{X}.\delta, \mathbf{X}:\mathbf{X}$. Note that
 - $\varepsilon' = (\forall \mathbf{X}.\delta, \mathbf{X}:\mathbf{X}, \text{inj}_{\forall}(\forall \mathbf{X}.\text{inj}_{?}), \forall \mathbf{X}.\text{inj}_{?})$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_{\mathbf{R}}$.
 - $\varepsilon'' = \varepsilon \circledast \varepsilon'$ is defined because

$$\begin{aligned} & [(\forall \mathbf{X}.\mathbf{G}_1, \forall \mathbf{X}.\mathbf{c}_{11}, \text{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_{21}))] \circledast [(\forall \mathbf{X}.\delta, \mathbf{X}:\mathbf{X}, \text{inj}_{\forall}(\forall \mathbf{X}.\text{inj}_{?}), \forall \mathbf{X}.\text{inj}_{?})] = \\ & \quad [(\forall \mathbf{X}.\mathbf{G}_1, \forall \mathbf{X}.\mathbf{c}_{11}, \forall \mathbf{X}.\mathbf{c}_{21})] \in \varepsilon'' \end{aligned}$$

by Lemma B.41. Thus, the result follows.

- $(\mathbf{G}_1 \rightarrow \mathbf{G}_2, \text{inj}_{\rightarrow}(\mathbf{c}_{11} \rightarrow \mathbf{c}_{12}), \text{inj}_{\rightarrow}(\mathbf{c}_{21} \rightarrow \mathbf{c}_{22})) \in \varepsilon$. Then, we can take $\mathbf{G}_{\mathbf{R}} = ?_{\delta} \rightarrow ?_{\delta}$. Note that

- $\varepsilon' = (?_{\delta} \rightarrow ?_{\delta}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_{\mathbf{R}}$.
- $\varepsilon'' = \varepsilon \circledast \varepsilon'$ is defined because

$$\begin{aligned} & [(\mathbf{G}_1 \rightarrow \mathbf{G}_2, \text{inj}_{\rightarrow}(\mathbf{c}_{11} \rightarrow \mathbf{c}_{12}), \text{inj}_{\rightarrow}(\mathbf{c}_{21} \rightarrow \mathbf{c}_{22}))] \circledast [(?_{\delta} \rightarrow ?_{\delta}, \text{inj}_{\rightarrow}(\text{inj}_{?} \rightarrow \text{inj}_{?}), \text{inj}_{?} \rightarrow \text{inj}_{?})] = \\ & \quad [(\mathbf{G}_1 \rightarrow \mathbf{G}_2, \text{inj}_{\rightarrow}(\mathbf{c}_{11} \rightarrow \mathbf{c}_{12}), \mathbf{c}_{21} \rightarrow \mathbf{c}_{22})] \in \varepsilon'' \end{aligned}$$

by Lemma B.41. Thus, the result follows.

- $(\mathbf{G}_1 \times \mathbf{G}_2, \text{inj}_{\times}(\mathbf{c}_{11} \times \mathbf{c}_{12}), \text{inj}_{\times}(\mathbf{c}_{21} \times \mathbf{c}_{22})) \in \varepsilon$. Then, we can take $\mathbf{G}_{\mathbf{R}} = ?_{\delta} \times ?_{\delta}$. Note that

- $\varepsilon' = (?_{\delta} \times ?_{\delta}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_{\mathbf{R}}$.
- $\varepsilon'' = \varepsilon \circledast \varepsilon'$ is defined because

$$\begin{aligned} & [(\mathbf{G}_1 \times \mathbf{G}_2, \text{inj}_{\times}(\mathbf{c}_{11} \times \mathbf{c}_{12}), \text{inj}_{\times}(\mathbf{c}_{21} \times \mathbf{c}_{22}))] \circledast [(?_{\delta} \times ?_{\delta}, \text{inj}_{\times}(\text{inj}_{?} \times \text{inj}_{?}), \text{inj}_{?} \times \text{inj}_{?})] = \\ & \quad [(\mathbf{G}_1 \times \mathbf{G}_2, \text{inj}_{\times}(\mathbf{c}_{11} \times \mathbf{c}_{12}), \mathbf{c}_{21} \times \mathbf{c}_{22})] \in \varepsilon'' \end{aligned}$$

by Lemma B.41. Thus, the result follows.

- $(\forall \mathbf{X}.\mathbf{G}_1, \text{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_{11}), \text{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_{21})) \in \varepsilon$. Then, we can take $\mathbf{G}_{\mathbf{R}} = \forall \mathbf{X}.\delta, \mathbf{X}:\mathbf{X}$. Note that
 - $\varepsilon' = (\forall \mathbf{X}.\delta, \mathbf{X}:\mathbf{X}, \text{inj}_{\forall}(\forall \mathbf{X}.\text{inj}_{?}), \forall \mathbf{X}.\text{inj}_{?})$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_{\mathbf{R}}$.
 - $\varepsilon'' = \varepsilon \circledast \varepsilon'$ is defined because

$$\begin{aligned} & [(\forall \mathbf{X}.\mathbf{G}_1, \text{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_{11}), \text{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_{21}))] \circledast [(\forall \mathbf{X}.\delta, \mathbf{X}:\mathbf{X}, \text{inj}_{\forall}(\forall \mathbf{X}.\text{inj}_{?}), \forall \mathbf{X}.\text{inj}_{?})] = \\ & \quad [(\forall \mathbf{X}.\mathbf{G}_1, \text{inj}_{\forall}(\forall \mathbf{X}.\mathbf{c}_{11}), \forall \mathbf{X}.\mathbf{c}_{21})] \in \varepsilon'' \end{aligned}$$

by Lemma B.41. Thus, the result follows.

- $(?_{\delta'}, \text{inj}_{?}, \text{inj}_{?}) \in \varepsilon$. Then, we can take $\mathbf{G}_{\mathbf{R}} = \mathbf{B}$. Note that

- $\varepsilon' = (\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_{\mathbf{R}}$.
- $\varepsilon'' = \varepsilon \circ \varepsilon'$ is defined because $[(\text{?}_{\delta'}, \text{inj}_{\text{?}}, \text{inj}_{\text{?}})] \circ [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})] = [(\mathbf{B}, \text{inj}_{\mathbf{B}}, \mathbf{B})] \in \varepsilon''$.
Thus, the result follows.

□

Lemma B.62 *If $\varepsilon : \mathbf{G} \sim \mathbf{G}'$ and $\varepsilon' = \{(\mathbf{G}', \text{refl}_{\mathbf{G}'}^{\sqsubseteq}, \text{refl}_{\mathbf{G}'}^{\sqsubseteq})\}$, then $\varepsilon \circ \varepsilon' = \varepsilon$.*

PROOF. We know that $\varepsilon' = [(\mathbf{G}', \text{refl}_{\mathbf{G}'}^{\sqsubseteq}, \text{refl}_{\mathbf{G}'}^{\sqsubseteq})]$. We are required to prove that $\varepsilon \circ \varepsilon' = \varepsilon$. Or what is the same, we are required to prove that for all $(\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2) \in \varepsilon$, we get that

$$[(\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2)] \circ [(\mathbf{G}', \text{refl}_{\mathbf{G}'}^{\sqsubseteq}, \text{refl}_{\mathbf{G}'}^{\sqsubseteq})] = [(\mathbf{G}'', \mathbf{c}_1, \mathbf{c}_2)]$$

Since $\mathbf{c}_2 : \mathbf{G}'' \sqsubseteq \mathbf{G}'$, by Lemma B.64, we know that $\text{pullback}(\mathbf{G}', (\mathbf{G}'', \mathbf{c}_2), (\mathbf{G}', \text{refl}_{\mathbf{G}'}^{\sqsubseteq})) = (\mathbf{G}'', \text{refl}_{\mathbf{G}''}^{\sqsubseteq}, \mathbf{c}_2, \mathbf{c}_2)$. Since $\mathbf{c}_1 : \mathbf{G}'' \sqsubseteq \mathbf{G}$ and $\mathbf{c}_2 : \mathbf{G}'' \sqsubseteq \mathbf{G}'$, by Lemma B.87, we get that $\text{refl}_{\mathbf{G}''}^{\sqsubseteq}; \mathbf{c}_1 = \mathbf{c}_1$ and $\mathbf{c}_2; \text{refl}_{\mathbf{G}'}^{\sqsubseteq} = \mathbf{c}_2$. Therefore, the result follows immediately. □

Lemma B.63 $\text{reflEv}(\mathbf{G}) = \text{initEv}(\mathbf{G}, \mathbf{G}) = \{(\mathbf{G}, \text{refl}_{\mathbf{G}}^{\sqsubseteq}, \text{refl}_{\mathbf{G}}^{\sqsubseteq})\}$.

PROOF. $\text{reflEv}(\mathbf{G}) = \text{initEv}(\mathbf{G}, \mathbf{G}) = [(\mathbf{G}, \text{refl}_{\mathbf{G}}^{\sqsubseteq}, \text{refl}_{\mathbf{G}}^{\sqsubseteq})]$. By definition of $\text{initEv}(\mathbf{G}, \mathbf{G})$ and Lemma B.114,

$$\text{initEv}(\mathbf{G}, \mathbf{G}) = [(\mathbf{G} \sqcap \mathbf{G}, \text{initPT}(\mathbf{G}, \mathbf{G}), \text{initPT}(\mathbf{G}, \mathbf{G}))] = [(\mathbf{G}, \text{refl}_{\mathbf{G}}^{\sqsubseteq}, \text{refl}_{\mathbf{G}}^{\sqsubseteq})]$$

Therefore, the result follows. □

Lemma B.64 *If $\mathbf{c} : \mathbf{G}' \sqsubseteq \mathbf{G}$ then $\text{pullback}(\mathbf{G}, (\mathbf{G}', \mathbf{c}), (\mathbf{G}, \text{refl}_{\mathbf{G}}^{\sqsubseteq})) = (\mathbf{G}', \text{refl}_{\mathbf{G}'}^{\sqsubseteq}, \mathbf{c}, \mathbf{c})$.*

PROOF. We proceed by induction on $\mathbf{c} : \mathbf{G}' \sqsubseteq \mathbf{G}$.

Case $(\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B})$. We know that $\mathbf{G} = \mathbf{G}' = \mathbf{B}$ and $\mathbf{c} = \mathbf{B}$. We know that $\text{refl}_{\mathbf{B}}^{\sqsubseteq} = \mathbf{B}$ and

$$\text{pullback}(\mathbf{G}, (\mathbf{G}', \mathbf{c}), (\mathbf{G}, \text{refl}_{\mathbf{G}}^{\sqsubseteq})) = \text{pullback}(\mathbf{B}, (\mathbf{B}, \mathbf{B}), (\mathbf{B}, \text{refl}_{\mathbf{B}}^{\sqsubseteq})) = (\mathbf{B}, \mathbf{B}, \mathbf{B}, \mathbf{B}) = (\mathbf{G}', \text{refl}_{\mathbf{G}'}^{\sqsubseteq}, \mathbf{c}, \mathbf{c})$$

Therefore, the result follows immediately.

Case $(\mathbf{X} : \mathbf{X} \sqsubseteq \mathbf{X})$. We know that $\mathbf{G} = \mathbf{G}' = \mathbf{X}$ and $\mathbf{c} = \mathbf{X}$. We know that $\text{refl}_{\mathbf{X}}^{\sqsubseteq} = \mathbf{X}$ and

$$\text{pullback}(\mathbf{G}, (\mathbf{G}', \mathbf{c}), (\mathbf{G}, \text{refl}_{\mathbf{G}}^{\sqsubseteq})) = \text{pullback}(\mathbf{X}, (\mathbf{X}, \mathbf{X}), (\mathbf{X}, \text{refl}_{\mathbf{X}}^{\sqsubseteq})) = (\mathbf{X}, \mathbf{X}, \mathbf{X}, \mathbf{X}) = (\mathbf{G}', \text{refl}_{\mathbf{G}'}^{\sqsubseteq}, \mathbf{c}, \mathbf{c})$$

Therefore, the result follows immediately.

Case $(\mathbf{c}_1 \rightarrow \mathbf{c}_2 : \mathbf{G}'_1 \rightarrow \mathbf{G}'_2 \sqsubseteq \mathbf{G}_1 \rightarrow \mathbf{G}_2)$. We know that $\mathbf{G}' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$, $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$, $\mathbf{c} = \mathbf{c}_1 \rightarrow \mathbf{c}_2$, $\text{refl}_{\mathbf{G}'_1 \rightarrow \mathbf{G}'_2}^{\sqsubseteq} = \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq} \rightarrow \text{refl}_{\mathbf{G}'_2}^{\sqsubseteq}$ and $\text{refl}_{\mathbf{G}_1 \rightarrow \mathbf{G}_2}^{\sqsubseteq} = \text{refl}_{\mathbf{G}_1}^{\sqsubseteq} \rightarrow \text{refl}_{\mathbf{G}_2}^{\sqsubseteq}$. Also we know that

$$\frac{\mathbf{c}_1 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}_1 \quad \mathbf{c}_2 : \mathbf{G}'_2 \sqsubseteq \mathbf{G}_2}{\mathbf{c}_1 \rightarrow \mathbf{c}_2 : \mathbf{G}'_1 \rightarrow \mathbf{G}'_2 \sqsubseteq \mathbf{G}_1 \rightarrow \mathbf{G}_2}$$

We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}', \mathbf{c}), (\mathbf{G}, \text{refl}_{\mathbf{G}}^{\sqsubseteq})) &= \text{pullback}(\mathbf{G}_1 \rightarrow \mathbf{G}_2, (\mathbf{G}'_1 \rightarrow \mathbf{G}'_2, \mathbf{c}_1 \rightarrow \mathbf{c}_2), (\mathbf{G}_1 \rightarrow \mathbf{G}_2, \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq} \rightarrow \text{refl}_{\mathbf{G}'_2}^{\sqsubseteq})) = \\ &(\mathbf{G}_3 \rightarrow \mathbf{G}'_3, \mathbf{c}_3 \rightarrow \mathbf{c}'_3, \mathbf{c}_4 \rightarrow \mathbf{c}'_4, \mathbf{c}_5 \rightarrow \mathbf{c}'_5) \end{aligned}$$

where $\text{pullback}(\mathbf{G}_1, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}_1, \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq})) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$ and $\text{pullback}(\mathbf{G}_2, (\mathbf{G}'_2, \mathbf{c}_2), (\mathbf{G}_2, \text{refl}_{\mathbf{G}'_2}^{\sqsubseteq})) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. By the induction hypothesis on $\mathbf{c}_1 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}_1$ and $\mathbf{c}_2 : \mathbf{G}'_2 \sqsubseteq \mathbf{G}_2$, we know that $\text{pullback}(\mathbf{G}_1, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}_1, \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq})) = (\mathbf{G}'_1, \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq}, \mathbf{c}_1, \mathbf{c}_1)$ and $\text{pullback}(\mathbf{G}_2, (\mathbf{G}'_2, \mathbf{c}_2), (\mathbf{G}_2, \text{refl}_{\mathbf{G}'_2}^{\sqsubseteq})) = (\mathbf{G}'_2, \text{refl}_{\mathbf{G}'_2}^{\sqsubseteq}, \mathbf{c}_2, \mathbf{c}_2)$. Therefore, the result follows immediately.

Case $(\mathbf{c}_1 \times \mathbf{c}_2 : \mathbf{G}'_1 \times \mathbf{G}'_2 \sqsubseteq \mathbf{G}_1 \times \mathbf{G}_2)$. We know that $\mathbf{G}' = \mathbf{G}'_1 \times \mathbf{G}'_2$, $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$, $\mathbf{c} = \mathbf{c}_1 \times \mathbf{c}_2$, $\text{refl}_{\mathbf{G}'_1 \times \mathbf{G}'_2}^{\sqsubseteq} = \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq} \times \text{refl}_{\mathbf{G}'_2}^{\sqsubseteq}$ and $\text{refl}_{\mathbf{G}_1 \times \mathbf{G}_2}^{\sqsubseteq} = \text{refl}_{\mathbf{G}_1}^{\sqsubseteq} \times \text{refl}_{\mathbf{G}_2}^{\sqsubseteq}$. Also we know that

$$\frac{\mathbf{c}_1 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}_1 \quad \mathbf{c}_2 : \mathbf{G}'_2 \sqsubseteq \mathbf{G}_2}{\mathbf{c}_1 \rightarrow \mathbf{c}_2 : \mathbf{G}'_1 \rightarrow \mathbf{G}'_2 \sqsubseteq \mathbf{G}_1 \rightarrow \mathbf{G}_2}$$

We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}', \mathbf{c}), (\mathbf{G}, \text{refl}_{\mathbf{G}}^{\sqsubseteq})) &= \text{pullback}(\mathbf{G}_1 \times \mathbf{G}_2, (\mathbf{G}'_1 \times \mathbf{G}'_2, \mathbf{c}_1 \times \mathbf{c}_2), (\mathbf{G}_1 \times \mathbf{G}_2, \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq} \times \text{refl}_{\mathbf{G}'_2}^{\sqsubseteq})) = \\ &(\mathbf{G}_3 \times \mathbf{G}'_3, \mathbf{c}_3 \times \mathbf{c}'_3, \mathbf{c}_4 \times \mathbf{c}'_4, \mathbf{c}_5 \times \mathbf{c}'_5) \end{aligned}$$

where $\text{pullback}(\mathbf{G}_1, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}_1, \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq})) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$ and $\text{pullback}(\mathbf{G}_2, (\mathbf{G}'_2, \mathbf{c}_2), (\mathbf{G}_2, \text{refl}_{\mathbf{G}'_2}^{\sqsubseteq})) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. By the induction hypothesis on $\mathbf{c}_1 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}_1$ and $\mathbf{c}_2 : \mathbf{G}'_2 \sqsubseteq \mathbf{G}_2$, we know that $\text{pullback}(\mathbf{G}_1, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}_1, \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq})) = (\mathbf{G}'_1, \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq}, \mathbf{c}_1, \mathbf{c}_1)$ and $\text{pullback}(\mathbf{G}_2, (\mathbf{G}'_2, \mathbf{c}_2), (\mathbf{G}_2, \text{refl}_{\mathbf{G}'_2}^{\sqsubseteq})) = (\mathbf{G}'_2, \text{refl}_{\mathbf{G}'_2}^{\sqsubseteq}, \mathbf{c}_2, \mathbf{c}_2)$. Therefore, the result follows immediately.

Case $(\forall \mathbf{X}. \mathbf{c}_1 : \forall \mathbf{X}. \mathbf{G}'_1 \sqsubseteq \forall \mathbf{X}. \mathbf{G}_1)$. We know that $\mathbf{G}' = \forall \mathbf{X}. \mathbf{G}'_1$, $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}_1$, $\mathbf{c} = \forall \mathbf{X}. \mathbf{c}_1$, $\text{refl}_{\forall \mathbf{X}. \mathbf{G}'_1}^{\sqsubseteq} = \forall \mathbf{X}. \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq}$ and $\text{refl}_{\forall \mathbf{X}. \mathbf{G}_1}^{\sqsubseteq} = \forall \mathbf{X}. \text{refl}_{\mathbf{G}_1}^{\sqsubseteq}$. Also we know that

$$\frac{\mathbf{c}_1 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}_1}{\forall \mathbf{X}. \mathbf{c}_1 : \forall \mathbf{X}. \mathbf{G}'_1 \sqsubseteq \forall \mathbf{X}. \mathbf{G}_1}$$

We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}', \mathbf{c}), (\mathbf{G}, \text{refl}_{\mathbf{G}}^{\sqsubseteq})) &= \text{pullback}(\forall \mathbf{X}. \mathbf{G}_1, (\forall \mathbf{X}. \mathbf{G}'_1, \forall \mathbf{X}. \mathbf{c}_1), (\forall \mathbf{X}. \mathbf{G}_1, \forall \mathbf{X}. \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq})) = \\ &(\forall \mathbf{X}. \mathbf{G}_3, \forall \mathbf{X}. \mathbf{c}_3, \forall \mathbf{X}. \mathbf{c}_4, \forall \mathbf{X}. \mathbf{c}_5) \end{aligned}$$

where $\text{pullback}(\mathbf{G}_1, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}_1, \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq})) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$. By the induction hypothesis on $\mathbf{c}_1 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}_1$, we know that $\text{pullback}(\mathbf{G}_1, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}_1, \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq})) = (\mathbf{G}'_1, \text{refl}_{\mathbf{G}'_1}^{\sqsubseteq}, \mathbf{c}_1, \mathbf{c}_1)$. Therefore, the result follows immediately.

Case $(\text{inj}_{\mathbf{X}} : \mathbf{F} \sqsubseteq ?_{\delta})$. We know that $\mathbf{G}' = \mathbf{F}$, $\mathbf{G} = ?_{\delta}$, $\mathbf{c} = \text{inj}_{\mathbf{X}}$, $\text{refl}_{?_{\delta}}^{\sqsubseteq} = \text{inj}_{?}$ and $\text{refl}_{\mathbf{F}}^{\sqsubseteq} = \mathbf{F}$. Also, we know that

$$\frac{\mathbf{X} : \mathbf{F} \in \delta}{\text{inj}_{\mathbf{X}} : \mathbf{F} \sqsubseteq ?_{\delta}}$$

We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}', \mathbf{c}), (\mathbf{G}, \text{refl}_{\mathbf{G}}^{\sqsubseteq})) = \text{pullback}(?_{\delta}, (\mathbf{F}, \text{inj}_{\mathbf{X}}), (?_{\delta}, \text{inj}_{?})) = (\mathbf{F}, \mathbf{F}, \text{inj}_{\mathbf{X}}, \text{inj}_{\mathbf{X}}) = (\mathbf{G}', \text{refl}_{\mathbf{G}'}^{\sqsubseteq}, \mathbf{c}, \mathbf{c})$$

Therefore, the result follows immediately.

Case ($\text{inj}_B : B \sqsubseteq ?_\delta$). We know that $G' = B$, $G = ?_\delta$, $c = \text{inj}_B$, $\text{refl}_{?_\delta}^{\sqsubseteq} = \text{inj}_?$ and $\text{refl}_B^{\sqsubseteq} = B$. Also, we know that

$$\overline{\text{inj}_B : B \sqsubseteq ?_\delta}$$

We know that

$$\text{pullback}(G, (G', c), (G, \text{refl}_G^{\sqsubseteq})) = \text{pullback}(?_\delta, (B, \text{inj}_B), (?_\delta, \text{inj}_?)) = (B, B, \text{inj}_B, \text{inj}_B) = (G', \text{refl}_{G'}^{\sqsubseteq}, c, c)$$

Therefore, the result follows immediately.

Case ($\text{inj}_\rightarrow(c_1) : G' \sqsubseteq ?_\delta$). We know that $G = ?_\delta$, $c = \text{inj}_\rightarrow(c_1)$ and $\text{refl}_{?_\delta}^{\sqsubseteq} = \text{inj}_?$. Also, we know that

$$\frac{c_1 : G' \sqsubseteq ?_\delta \rightarrow ?_\delta}{\text{inj}_\rightarrow(c_1) : G' \sqsubseteq ?_\delta}$$

We know that

$$\text{pullback}(G, (G', c), (G, \text{refl}_G^{\sqsubseteq})) = \text{pullback}(?_\delta, (G', \text{inj}_\rightarrow(c_1)), (?_\delta, \text{inj}_?)) = (G_3, c_4, \text{inj}_\rightarrow(c_3), \text{inj}_\rightarrow(c_5))$$

where $\text{pullback}(?_\delta \rightarrow ?_\delta, (?_\delta \rightarrow ?_\delta, \text{inj}_? \rightarrow \text{inj}_?), (G', c_1)) = (G_3, c_3, c_4, c_5)$. We are required to prove that

$$\text{pullback}(G, (G', c), (G, \text{refl}_G^{\sqsubseteq})) = \text{pullback}(?_\delta, (G', \text{inj}_\rightarrow(c_1)), (?_\delta, \text{inj}_?)) = (G', \text{refl}_{G'}^{\sqsubseteq}, \text{inj}_\rightarrow(c_1), \text{inj}_\rightarrow(c_1))$$

By the induction hypothesis on $c_1 : G' \sqsubseteq ?_\delta \rightarrow ?_\delta$, we get that

$$\text{pullback}(?_\delta \rightarrow ?_\delta, (G', c_1), (?_\delta \rightarrow ?_\delta, \text{inj}_? \rightarrow \text{inj}_?)) = (G', \text{refl}_{G'}^{\sqsubseteq}, c_1, c_1)$$

Thus, we get that

$$\text{pullback}(?_\delta \rightarrow ?_\delta, (?_\delta \rightarrow ?_\delta, \text{inj}_? \rightarrow \text{inj}_?), (G', c_1)) = (G', c_1, \text{refl}_{G'}^{\sqsubseteq}, c_1)$$

Therefore, the result follows immediately.

Case ($\text{inj}_\times(c_1) : G' \sqsubseteq ?_\delta$). We know that $G = ?_\delta$, $c = \text{inj}_\times(c_1)$ and $\text{refl}_{?_\delta}^{\sqsubseteq} = \text{inj}_?$. Also, we know that

$$\frac{c_1 : G' \sqsubseteq ?_\delta \times ?_\delta}{\text{inj}_\times(c_1) : G' \sqsubseteq ?_\delta}$$

We know that

$$\text{pullback}(G, (G', c), (G, \text{refl}_G^{\sqsubseteq})) = \text{pullback}(?_\delta, (G', \text{inj}_\times(c_1)), (?_\delta, \text{inj}_?)) = (G_3, c_4, \text{inj}_\times(c_3), \text{inj}_\times(c_5))$$

where $\text{pullback}(?_\delta \times ?_\delta, (?_\delta \times ?_\delta, \text{inj}_? \times \text{inj}_?), (G', c_1)) = (G_3, c_3, c_4, c_5)$. We are required to prove that

$$\text{pullback}(G, (G', c), (G, \text{refl}_G^{\sqsubseteq})) = \text{pullback}(?_\delta, (G', \text{inj}_\times(c_1)), (?_\delta, \text{inj}_?)) = (G', \text{refl}_{G'}^{\sqsubseteq}, \text{inj}_\times(c_1), \text{inj}_\times(c_1))$$

By the induction hypothesis on $c_1 : G' \sqsubseteq ?_\delta \times ?_\delta$, we get that

$$\text{pullback}(?_\delta \times ?_\delta, (G', c_1), (?_\delta \times ?_\delta, \text{inj}_? \times \text{inj}_?)) = (G', \text{refl}_{G'}^{\sqsubseteq}, c_1, c_1)$$

Thus, we get that

$$\text{pullback}(?_\delta \times ?_\delta, (?_\delta \times ?_\delta, \text{inj}_? \times \text{inj}_?), (G', c_1)) = (G', c_1, \text{refl}_{G'}^{\sqsubseteq}, c_1)$$

Therefore, the result follows immediately.

Case $(\text{inj}_\forall(\mathbf{c}_1) : \mathbf{G}' \sqsubseteq ?_\delta)$. We know that $\mathbf{G} = ?_\delta$, $\mathbf{c} = \text{inj}_\forall(\mathbf{c}_1)$ and $\text{refl}_{?_\delta}^\sqsubseteq = \text{inj}_?$. Also, we know that

$$\frac{\mathbf{c}_1 : \mathbf{G}' \sqsubseteq \forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}}{\text{inj}_\forall(\mathbf{c}_1) : \mathbf{G}' \sqsubseteq ?_\delta}$$

We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}', \mathbf{c}), (\mathbf{G}, \text{refl}_{\mathbf{G}}^\sqsubseteq)) = \text{pullback}(?_\delta, (\mathbf{G}', \text{inj}_\forall(\mathbf{c}_1)), (?_\delta, \text{inj}_?)) = (\mathbf{G}_3, \mathbf{c}_4, \text{inj}_\forall(\mathbf{c}_3), \text{inj}_\forall(\mathbf{c}_5))$$

where $\text{pullback}(\forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}, (\forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}, \forall \mathbf{X}. \text{inj}_?), (\mathbf{G}', \mathbf{c}_1)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5)$. We are required to prove that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}', \mathbf{c}), (\mathbf{G}, \text{refl}_{\mathbf{G}}^\sqsubseteq)) = \text{pullback}(?_\delta, (\mathbf{G}', \text{inj}_\forall(\mathbf{c}_1)), (?_\delta, \text{inj}_?)) = (\mathbf{G}', \text{refl}_{\mathbf{G}'}^\sqsubseteq, \text{inj}_\forall(\mathbf{c}_1), \text{inj}_\forall(\mathbf{c}_1))$$

By the induction hypothesis on $\mathbf{c}_1 : \mathbf{G}' \sqsubseteq \forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}$, we get that

$$\text{pullback}(\forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}, (\mathbf{G}', \mathbf{c}_1), (\forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}, \forall \mathbf{X}. \text{inj}_?)) = (\mathbf{G}', \text{refl}_{\mathbf{G}'}^\sqsubseteq, \mathbf{c}_1, \mathbf{c}_1)$$

Thus, we get that

$$\text{pullback}(\forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}, (\forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}, \forall \mathbf{X}. \text{inj}_?), (\mathbf{G}', \mathbf{c}_1)) = (\mathbf{G}', \mathbf{c}_1, \text{refl}_{\mathbf{G}'}^\sqsubseteq, \mathbf{c}_1)$$

Therefore, the result follows immediately.

Case $(\text{inj}_? : ?_\delta \sqsubseteq ?_{\delta'})$. We know that $\mathbf{G}' = ?_{\delta'}$, $\mathbf{G} = ?_\delta$, $\mathbf{c} = \text{inj}_?$, $\text{refl}_{?_{\delta'}}^\sqsubseteq = \text{inj}_?$ and $\text{refl}_{?_\delta}^\sqsubseteq = \text{inj}_?$. Also, we know that

$$\frac{\delta' \subseteq \delta}{\text{inj}_? : ?_{\delta'} \sqsubseteq ?_\delta}$$

We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}', \mathbf{c}), (\mathbf{G}, \text{refl}_{\mathbf{G}}^\sqsubseteq)) &= \text{pullback}(?_\delta, (?_{\delta'}, \text{inj}_?), (?_\delta, \text{inj}_?)) = \\ & (?_{\delta'}, \text{inj}_?, \text{inj}_?, \text{inj}_?) = (\mathbf{G}', \text{refl}_{\mathbf{G}'}^\sqsubseteq, \mathbf{c}, \mathbf{c}) \end{aligned}$$

Therefore, the result follows immediately. □

Lemma B.65 *If $\varepsilon : \mathbf{G} \sim ?_\delta$, $\mathbf{G} \not\rightsquigarrow ?_{\delta'}$, $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$ and $\varepsilon \circ \varepsilon'$ is defined, then $\mathcal{S}(\mathbf{G}, ?_\delta) = \mathbf{G}_R$.*

PROOF. We follow by case analysis on \mathbf{G}_R .

Case $(\mathbf{G}_R = \mathbf{B})$. We know that $\varepsilon' = [(\mathbf{B}, \text{inj}_\mathbf{B}, \mathbf{B})]$ or $\varepsilon' = [(\mathbf{B}, \text{inj}_\mathbf{X}, \mathbf{B})]$ ($\mathbf{X} : \mathbf{B} \in \delta$), where $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$. Therefore, $\varepsilon' : ?_\delta \sim \mathbf{B}$ and $\mathbf{G}_R = \mathbf{B}$. Since $\varepsilon \circ \varepsilon'$ is defined, we know that $\varepsilon \circ \varepsilon' : \mathbf{G} \sim \mathbf{B}$. Since $\mathbf{G} \not\rightsquigarrow ?_{\delta'}$, then $\mathbf{G} = \mathbf{B}$. Therefore, $\mathcal{S}(\mathbf{G}, ?_\delta) = \mathcal{S}(\mathbf{B}, ?_\delta) = \mathbf{B} = \mathbf{G}_R$, and the result follows immediately.

Case $(\mathbf{G}_R = \mathbf{F})$. Therefore, $\varepsilon' = [(\mathbf{F}, \text{inj}_\mathbf{X}, \mathbf{F})]$, where $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$ and $\mathbf{X} : \mathbf{F} \in \delta$. Therefore, $\varepsilon' : ?_\delta \sim \mathbf{F}$. Since $\varepsilon \circ \varepsilon'$ is defined, we know that $\varepsilon \circ \varepsilon' : \mathbf{G} \sim \mathbf{F}$. Since $\mathbf{G} \not\rightsquigarrow ?_{\delta'}$, then $\mathbf{G} = \mathbf{F}$. Therefore, $\mathcal{S}(\mathbf{G}, ?_\delta) = \mathcal{S}(\mathbf{F}, ?_\delta) = \mathbf{F} = \mathbf{G}_R$, and the result follows immediately.

Case $(\mathbf{G}_R = ?_\delta \rightarrow ?_\delta)$. We know that $\varepsilon' = [(?_\delta \rightarrow ?_\delta, \text{inj}_\rightarrow(\text{inj}_? \rightarrow \text{inj}_?), \text{inj}_? \rightarrow \text{inj}_?)]$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$. Since $\varepsilon \circ \varepsilon'$ is defined, we know that $\varepsilon \circ \varepsilon' : \mathbf{G} \sim ?_\delta \rightarrow ?_\delta$. Since $\mathbf{G} \not\rightsquigarrow ?_{\delta'}$, then $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$, for some \mathbf{G}_1 and \mathbf{G}_2 . Therefore, $\mathcal{S}(\mathbf{G}, ?_\delta) = \mathcal{S}(\mathbf{G}_1 \rightarrow \mathbf{G}_2, ?_\delta) = ?_\delta \rightarrow ?_\delta = \mathbf{G}_R$, and the result follows immediately.

Case ($\mathbf{G}_R = ?_\delta \times ?_\delta$). We know that $\varepsilon' = [(?_\delta \times ?_\delta, \text{inj}_\times(\text{inj}_? \times \text{inj}_?), \text{inj}_? \times \text{inj}_?)]$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$. Since $\varepsilon \circ \varepsilon'$ is defined, we know that $\varepsilon \circ \varepsilon' : \mathbf{G} \sim ?_\delta \times ?_\delta$. Since $\mathbf{G} \not\sim ?_{\delta'}$, then $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$, for some \mathbf{G}_1 and \mathbf{G}_2 . Therefore, $\mathcal{S}(\mathbf{G}, ?_\delta) = \mathcal{S}(\mathbf{G}_1 \times \mathbf{G}_2, ?_\delta) = ?_\delta \times ?_\delta = \mathbf{G}_R$, and the result follows immediately.

Case ($\mathbf{G}_R = \forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}$). We know that $\varepsilon' = [(\forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}, \text{inj}_\forall(\forall \mathbf{X}. \text{inj}_?), \forall \mathbf{X}. \text{inj}_?)]$ and $\vdash \varepsilon' : \delta \rightarrow \mathbf{G}_R$. Since $\varepsilon \circ \varepsilon'$ is defined, we know that $\varepsilon \circ \varepsilon' : \mathbf{G} \sim \forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X}$. Since $\mathbf{G} \not\sim ?_{\delta'}$, then $\mathbf{G} = \forall \mathbf{X}. \mathbf{G}_1$, for some \mathbf{G}_1 . Therefore, $\mathcal{S}(\mathbf{G}, ?_\delta) = \mathcal{S}(\forall \mathbf{X}. \mathbf{G}_1, ?_\delta) = \forall \mathbf{X}. ?_\delta, \mathbf{X} : \mathbf{X} = \mathbf{G}_R$, and the result follows immediately.

□

Lemma B.66 $\mathbf{c}; \mathbf{c}' = \mathbf{c}'' \iff \mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}]$.

PROOF. We are required to prove that $\mathbf{c}; \mathbf{c}' = \mathbf{c}'' \iff \mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}]$. We first prove the \implies direction: $\mathbf{c}; \mathbf{c}' = \mathbf{c}'' \implies \mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}]$. We proceed by induction on $\mathbf{c}; \mathbf{c}' = \mathbf{c}''$.

Case ($\mathbf{B}; \mathbf{B} = \mathbf{B}$). We know that

- $\mathbf{c} = \mathbf{B}$, $\mathbf{c}' = \mathbf{B}$ and $\mathbf{c}'' = \mathbf{B}$.
- $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}$.
- $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}$.
- $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}$.
- $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}]$, since $\mathbf{B}; \mathbf{B} = \mathbf{B}$.

Thus the result follows immediately.

Case ($\mathbf{Y}; \mathbf{Y} = \mathbf{Y}$). We have the following cases.

- $\mathbf{Y} = \mathbf{X}$. We know that
 - $\mathbf{c} = \mathbf{X}$, $\mathbf{c}' = \mathbf{X}$ and $\mathbf{c}'' = \mathbf{X}$.
 - $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{X}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{F}$.
 - $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{X}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{F}$.
 - $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{X}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{F}$.
 - $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}]$, since $\mathbf{F}; \mathbf{F} = \mathbf{F}$.

Thus the result follows immediately.

- $\mathbf{Y} \neq \mathbf{X}$. We know that
 - $\mathbf{c} = \mathbf{Y}$, $\mathbf{c}' = \mathbf{Y}$ and $\mathbf{c}'' = \mathbf{Y}$.
 - $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}$.
 - $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}$.
 - $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}$.

– $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}]$, since $\mathbf{Y}; \mathbf{Y} = \mathbf{Y}$.

Thus the result follows immediately.

Case $((\mathbf{c}_1 \rightarrow \mathbf{c}_2); (\mathbf{c}_1 \rightarrow \mathbf{c}'_2) = (\mathbf{c}_1; \mathbf{c}_1) \rightarrow (\mathbf{c}_2; \mathbf{c}'_2))$. We know that

- $\mathbf{c} = \mathbf{c}_1 \rightarrow \mathbf{c}_2$, $\mathbf{c}' = \mathbf{c}_1 \rightarrow \mathbf{c}'_2$ and $\mathbf{c}'' = (\mathbf{c}_1; \mathbf{c}_1) \rightarrow (\mathbf{c}_2; \mathbf{c}'_2)$.
- $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{c}_1 \rightarrow \mathbf{c}_2)[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] \rightarrow \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}]$.
- $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{c}_1 \rightarrow \mathbf{c}'_2)[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] \rightarrow \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}]$.
- $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] = ((\mathbf{c}_1; \mathbf{c}_1) \rightarrow (\mathbf{c}_2; \mathbf{c}'_2))[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{c}_1; \mathbf{c}_1)[\mathbf{X} \mapsto \mathbf{F}] \rightarrow (\mathbf{c}_2; \mathbf{c}'_2)[\mathbf{X} \mapsto \mathbf{F}]$.
- Thus, we are required to prove that

$$(\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] \rightarrow \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}]); (\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] \rightarrow \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}]) = (\mathbf{c}_1; \mathbf{c}_1)[\mathbf{X} \mapsto \mathbf{F}] \rightarrow (\mathbf{c}_2; \mathbf{c}'_2)[\mathbf{X} \mapsto \mathbf{F}]$$

Let us suppose that $\mathbf{c}_1; \mathbf{c}_1 = \mathbf{c}_1''$ and $\mathbf{c}_2; \mathbf{c}'_2 = \mathbf{c}_2''$, for some \mathbf{c}_1'' and \mathbf{c}_2'' . Therefore, we are required to show that $\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_1''[\mathbf{X} \mapsto \mathbf{F}]$ and $\mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_2''[\mathbf{X} \mapsto \mathbf{F}]$.

- By the induction hypothesis we get that $\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_1''[\mathbf{X} \mapsto \mathbf{F}]$ and $\mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_2''[\mathbf{X} \mapsto \mathbf{F}]$, as we required to show.

Case $((\mathbf{c}_1 \times \mathbf{c}_2); (\mathbf{c}_1 \times \mathbf{c}'_2) = (\mathbf{c}_1; \mathbf{c}_1) \times (\mathbf{c}_2; \mathbf{c}'_2))$. We know that

- $\mathbf{c} = \mathbf{c}_1 \times \mathbf{c}_2$, $\mathbf{c}' = \mathbf{c}_1 \times \mathbf{c}'_2$ and $\mathbf{c}'' = (\mathbf{c}_1; \mathbf{c}_1) \times (\mathbf{c}_2; \mathbf{c}'_2)$.
- $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{c}_1 \times \mathbf{c}_2)[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] \times \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}]$.
- $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{c}_1 \times \mathbf{c}'_2)[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] \times \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}]$.
- $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] = ((\mathbf{c}_1; \mathbf{c}_1) \times (\mathbf{c}_2; \mathbf{c}'_2))[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{c}_1; \mathbf{c}_1)[\mathbf{X} \mapsto \mathbf{F}] \times (\mathbf{c}_2; \mathbf{c}'_2)[\mathbf{X} \mapsto \mathbf{F}]$.
- Thus, we are required to prove that

$$(\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] \times \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}]); (\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] \times \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}]) = (\mathbf{c}_1; \mathbf{c}_1)[\mathbf{X} \mapsto \mathbf{F}] \times (\mathbf{c}_2; \mathbf{c}'_2)[\mathbf{X} \mapsto \mathbf{F}]$$

Let us suppose that $\mathbf{c}_1; \mathbf{c}_1 = \mathbf{c}_1''$ and $\mathbf{c}_2; \mathbf{c}'_2 = \mathbf{c}_2''$, for some \mathbf{c}_1'' and \mathbf{c}_2'' . Therefore, we are required to show that $\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_1''[\mathbf{X} \mapsto \mathbf{F}]$ and $\mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_2''[\mathbf{X} \mapsto \mathbf{F}]$.

- By the induction hypothesis we get that $\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_1''[\mathbf{X} \mapsto \mathbf{F}]$ and $\mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_2''[\mathbf{X} \mapsto \mathbf{F}]$, as we required to show.

Case $((\forall \mathbf{Y}. \mathbf{c}_1); (\forall \mathbf{Y}. \mathbf{c}_1) = \forall \mathbf{Y}. (\mathbf{c}_1; \mathbf{c}_1))$. We know that

- $\mathbf{c} = \forall \mathbf{Y}. \mathbf{c}_1$, $\mathbf{c}' = \forall \mathbf{Y}. \mathbf{c}_1$ and $\mathbf{c}'' = \forall \mathbf{Y}. (\mathbf{c}_1; \mathbf{c}_1)$.
- $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = (\forall \mathbf{Y}. \mathbf{c}_1)[\mathbf{X} \mapsto \mathbf{F}] = \forall \mathbf{Y}. \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]$.
- $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = (\forall \mathbf{Y}. \mathbf{c}_1)[\mathbf{X} \mapsto \mathbf{F}] = \forall \mathbf{Y}. \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]$.
- $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] = (\forall \mathbf{Y}. (\mathbf{c}_1; \mathbf{c}_1))[\mathbf{X} \mapsto \mathbf{F}] = \forall \mathbf{Y}. (\mathbf{c}_1; \mathbf{c}_1)[\mathbf{X} \mapsto \mathbf{F}]$.

- Thus, we are required to prove that

$$(\forall \mathbf{Y}. \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]); (\forall \mathbf{Y}. \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]) = \forall \mathbf{Y}. (\mathbf{c}_1; \mathbf{c}_1)[\mathbf{X} \mapsto \mathbf{F}]$$

Let us suppose that $\mathbf{c}_1; \mathbf{c}_1 = \mathbf{c}_1''$, for some \mathbf{c}_1'' . Therefore, we are required to show that $\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_1''[\mathbf{X} \mapsto \mathbf{F}]$.

- By the induction hypothesis we get that $\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_1''[\mathbf{X} \mapsto \mathbf{F}]$, as we required to show.

Case ($\mathbf{B}; \text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$). We know that

- $\mathbf{c} = \mathbf{B}$, $\mathbf{c}' = \text{inj}_{\mathbf{B}}$ and $\mathbf{c}'' = \text{inj}_{\mathbf{B}}$.
- $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}$.
- $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_{\mathbf{B}}[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_{\mathbf{B}}$.
- $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_{\mathbf{B}}[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_{\mathbf{B}}$.
- $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}]$, since $\mathbf{B}; \text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$.

Thus the result follows immediately.

Case ($\mathbf{F}'; \text{inj}_{\mathbf{Y}} = \text{inj}_{\mathbf{Y}}$). We have the following cases.

- $\mathbf{F}' = \mathbf{X}$. We know that
 - $\mathbf{c} = \mathbf{X}$, $\mathbf{c}' = \text{inj}_{\mathbf{Y}}$ and $\mathbf{c}'' = \text{inj}_{\mathbf{Y}}$.
 - $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{X}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{F}$.
 - $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_{\mathbf{Y}}[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_{\mathbf{Y}}$.
 - $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_{\mathbf{Y}}[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_{\mathbf{Y}}$.
 - $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}]$, since $\mathbf{F}; \text{inj}_{\mathbf{Y}} = \text{inj}_{\mathbf{Y}}$.

Thus the result follows immediately.

- $\mathbf{F}' \neq \mathbf{X}$. We know that
 - $\mathbf{c} = \mathbf{Y}$, $\mathbf{c}' = \mathbf{Y}$ and $\mathbf{c}'' = \mathbf{Y}$.
 - $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}$.
 - $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}$.
 - $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{Y}$.
 - $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}]$, since $\mathbf{Y}; \mathbf{Y} = \mathbf{Y}$.

Thus the result follows immediately.

Case ($\mathbf{c}; \text{inj}_-(\mathbf{c}_1) = \text{inj}_-(\mathbf{c}; \mathbf{c}_1)$). We know that

- $\mathbf{c}' = \text{inj}_-(\mathbf{c}_1)$ and $\mathbf{c}'' = \text{inj}_-(\mathbf{c}; \mathbf{c}_1)$.
- $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_-(\mathbf{c}_1)[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_-(\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}])$.

- $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_-(\mathbf{c}; \mathbf{c}_1)[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_-(\mathbf{c}; \mathbf{c}_1)[\mathbf{X} \mapsto \mathbf{F}]$.
- We are required to prove that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] = (\mathbf{c}; \mathbf{c}_1)[\mathbf{X} \mapsto \mathbf{F}]$, which follows immediately by the induction hypothesis.

Case $(\mathbf{c}; \text{inj}_? = \mathbf{c})$. We know that

- $\mathbf{c}' = \text{inj}_?$ and $\mathbf{c}'' = \mathbf{c}$.
- $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_?[\mathbf{X} \mapsto \mathbf{F}] = \text{inj}_?$.
- $\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]$.
- $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}]$, since $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]; \text{inj}_? = \mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]$.

Thus the result follows immediately.

The \Leftarrow is analogous. □

Lemma B.67 $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \iff \text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = (\mathbf{G}_3, \mathbf{c}_4, \mathbf{c}_3, \mathbf{c}_5)$.

PROOF. We are required to prove that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \iff \text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = (\mathbf{G}_3, \mathbf{c}_4, \mathbf{c}_3, \mathbf{c}_5)$$

We first prove the \implies direction:

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}_3, \mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5) \implies \text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = (\mathbf{G}_3, \mathbf{c}_4, \mathbf{c}_3, \mathbf{c}_5)$$

We proceed by induction on \mathbf{c}_1 and \mathbf{c}_2 and case analysis on $\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2))$.

Case $(\text{pullback}(\mathbf{B}, (\mathbf{B}, \mathbf{B}), (\mathbf{B}, \mathbf{B})))$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\mathbf{B}, (\mathbf{B}, \mathbf{B}), (\mathbf{B}, \mathbf{B})) = (\mathbf{B}, \mathbf{B}, \mathbf{B}, \mathbf{B}) = \text{pullback}(\mathbf{B}, (\mathbf{B}, \mathbf{B}), (\mathbf{B}, \mathbf{B})) =$$

Therefore, the results follows immediately.

Case $(\text{pullback}(\mathbf{X}, (\mathbf{X}, \mathbf{X}), (\mathbf{X}, \mathbf{X})))$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\mathbf{X}, (\mathbf{X}, \mathbf{X}), (\mathbf{X}, \mathbf{X})) = (\mathbf{X}, \mathbf{X}, \mathbf{X}, \mathbf{X}) = \text{pullback}(\mathbf{X}, (\mathbf{X}, \mathbf{X}), (\mathbf{X}, \mathbf{X})) =$$

Therefore, the results follows immediately.

Case $(\text{pullback}(\mathbf{G}'_1 \rightarrow \mathbf{G}'_2, (\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}), (\mathbf{G}_{21} \rightarrow \mathbf{G}_{22}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22})))$. We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\mathbf{G}'_1 \rightarrow \mathbf{G}'_2, (\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}), (\mathbf{G}_{21} \rightarrow \mathbf{G}_{22}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22})) = \\ &(\mathbf{G}'_3 \rightarrow \mathbf{G}''_3, \mathbf{c}'_3 \rightarrow \mathbf{c}''_3, \mathbf{c}'_4 \rightarrow \mathbf{c}''_4, \mathbf{c}'_5 \rightarrow \mathbf{c}''_5) \end{aligned}$$

where $\text{pullback}(\mathbf{G}'_1, (\mathbf{G}_{11}, \mathbf{c}_{11}), (\mathbf{G}_{21}, \mathbf{c}_{21})) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$ and $\text{pullback}(\mathbf{G}'_2, (\mathbf{G}_{12}, \mathbf{c}_{12}), (\mathbf{G}_{22}, \mathbf{c}_{22})) = (\mathbf{G}''_3, \mathbf{c}''_3, \mathbf{c}''_4, \mathbf{c}''_5)$. By the induction hypothesis we get that $\text{pullback}(\mathbf{G}'_1, (\mathbf{G}_{21}, \mathbf{c}_{21}), (\mathbf{G}_{11}, \mathbf{c}_{11})) = (\mathbf{G}'_3, \mathbf{c}'_4, \mathbf{c}'_3, \mathbf{c}'_5)$ and $\text{pullback}(\mathbf{G}'_2, (\mathbf{G}_{22}, \mathbf{c}_{22}), (\mathbf{G}_{12}, \mathbf{c}_{12})) = (\mathbf{G}''_3, \mathbf{c}''_4, \mathbf{c}''_3, \mathbf{c}''_5)$. Therefore, we have that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) &= \text{pullback}(\mathbf{G}'_1 \rightarrow \mathbf{G}'_2, (\mathbf{G}_{21} \rightarrow \mathbf{G}_{22}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22}), (\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12})) = \\ &(\mathbf{G}'_3 \rightarrow \mathbf{G}''_3, \mathbf{c}'_4 \rightarrow \mathbf{c}''_4, \mathbf{c}'_3 \rightarrow \mathbf{c}''_3, \mathbf{c}'_5 \rightarrow \mathbf{c}''_5) \end{aligned}$$

Finally, the results follows immediately.

Case ($\text{pullback}(\mathbf{G}'_1 \times \mathbf{G}'_2, (\mathbf{G}_{11} \times \mathbf{G}_{12}, \mathbf{c}_{11} \times \mathbf{c}_{12}), (\mathbf{G}_{21} \times \mathbf{G}_{22}, \mathbf{c}_{21} \times \mathbf{c}_{22}))$). We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\mathbf{G}'_1 \times \mathbf{G}'_2, (\mathbf{G}_{11} \times \mathbf{G}_{12}, \mathbf{c}_{11} \times \mathbf{c}_{12}), (\mathbf{G}_{21} \times \mathbf{G}_{22}, \mathbf{c}_{21} \times \mathbf{c}_{22})) = \\ &= (\mathbf{G}'_3 \times \mathbf{G}''_3, \mathbf{c}'_3 \times \mathbf{c}''_3, \mathbf{c}'_4 \times \mathbf{c}''_4, \mathbf{c}'_5 \times \mathbf{c}''_5) \end{aligned}$$

where $\text{pullback}(\mathbf{G}'_1, (\mathbf{G}_{11}, \mathbf{c}_{11}), (\mathbf{G}_{21}, \mathbf{c}_{21})) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$ and $\text{pullback}(\mathbf{G}'_2, (\mathbf{G}_{12}, \mathbf{c}_{12}), (\mathbf{G}_{22}, \mathbf{c}_{22})) = (\mathbf{G}''_3, \mathbf{c}''_3, \mathbf{c}''_4, \mathbf{c}''_5)$. By the induction hypothesis we get that $\text{pullback}(\mathbf{G}'_1, (\mathbf{G}_{21}, \mathbf{c}_{21}), (\mathbf{G}_{11}, \mathbf{c}_{11})) = (\mathbf{G}'_3, \mathbf{c}'_4, \mathbf{c}'_3, \mathbf{c}'_5)$ and $\text{pullback}(\mathbf{G}'_2, (\mathbf{G}_{22}, \mathbf{c}_{22}), (\mathbf{G}_{12}, \mathbf{c}_{12})) = (\mathbf{G}''_3, \mathbf{c}''_4, \mathbf{c}''_3, \mathbf{c}''_5)$. Therefore, we have that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) &= \text{pullback}(\mathbf{G}'_1 \times \mathbf{G}'_2, (\mathbf{G}_{21} \times \mathbf{G}_{22}, \mathbf{c}_{21} \times \mathbf{c}_{22}), (\mathbf{G}_{11} \times \mathbf{G}_{12}, \mathbf{c}_{11} \times \mathbf{c}_{12})) = \\ &= (\mathbf{G}'_3 \times \mathbf{G}''_3, \mathbf{c}'_4 \times \mathbf{c}''_4, \mathbf{c}'_3 \times \mathbf{c}''_3, \mathbf{c}'_5 \times \mathbf{c}''_5) \end{aligned}$$

Finally, the results follows immediately.

Case ($\text{pullback}(\forall \mathbf{X}. \mathbf{G}'_1, (\forall \mathbf{X}. \mathbf{G}_{11}, \forall \mathbf{X}. \mathbf{c}_{11}), (\forall \mathbf{X}. \mathbf{G}_{21}, \forall \mathbf{X}. \mathbf{c}_{21}))$). We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\forall \mathbf{X}. \mathbf{G}'_1, (\forall \mathbf{X}. \mathbf{G}_{11}, \forall \mathbf{X}. \mathbf{c}_{11}), (\forall \mathbf{X}. \mathbf{G}_{21}, \forall \mathbf{X}. \mathbf{c}_{21})) = \\ &= (\forall \mathbf{X}. \mathbf{G}'_3, \forall \mathbf{X}. \mathbf{c}'_3, \forall \mathbf{X}. \mathbf{c}'_4, \forall \mathbf{X}. \mathbf{c}'_5) \end{aligned}$$

where $\text{pullback}(\mathbf{G}'_1, (\mathbf{G}_{11}, \mathbf{c}_{11}), (\mathbf{G}_{21}, \mathbf{c}_{21})) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$ By the induction hypothesis we get that $\text{pullback}(\mathbf{G}'_1, (\mathbf{G}_{21}, \mathbf{c}_{21}), (\mathbf{G}_{11}, \mathbf{c}_{11})) = (\mathbf{G}'_3, \mathbf{c}'_4, \mathbf{c}'_3, \mathbf{c}'_5)$. Therefore, we have that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) &= \text{pullback}(\forall \mathbf{X}. \mathbf{G}'_1, (\forall \mathbf{X}. \mathbf{G}_{21}, \forall \mathbf{X}. \mathbf{c}_{21}), (\forall \mathbf{X}. \mathbf{G}_{11}, \forall \mathbf{X}. \mathbf{c}_{11})) = \\ &= (\forall \mathbf{X}. \mathbf{G}'_3, \forall \mathbf{X}. \mathbf{c}'_4, \forall \mathbf{X}. \mathbf{c}'_3, \forall \mathbf{X}. \mathbf{c}'_5) \end{aligned}$$

Finally, the results follows immediately.

Case ($\text{pullback}(\text{?}_\delta, (\mathbf{B}, \text{inj}_B), (\mathbf{B}, \text{inj}_B))$). We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\text{?}_\delta, (\mathbf{B}, \text{inj}_B), (\mathbf{B}, \text{inj}_B)) = (\mathbf{B}, \mathbf{B}, \mathbf{B}, \text{inj}_B) = \\ &= \text{pullback}(\text{?}_\delta, (\mathbf{B}, \text{inj}_B), (\mathbf{B}, \text{inj}_B)) = \text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) \end{aligned}$$

Therefore, the results follows immediately.

Case ($\text{pullback}(\text{?}_\delta, (\mathbf{F}, \text{inj}_X), (\mathbf{F}, \text{inj}_X))$). We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\text{?}_\delta, (\mathbf{F}, \text{inj}_X), (\mathbf{F}, \text{inj}_X)) = (\mathbf{F}, \mathbf{F}, \mathbf{F}, \text{inj}_X) = \\ &= \text{pullback}(\text{?}_\delta, (\mathbf{F}, \text{inj}_X), (\mathbf{F}, \text{inj}_X)) = \text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) \end{aligned}$$

Therefore, the results follows immediately.

Case ($\text{pullback}(\text{?}_\delta, (\mathbf{G}'_1, \text{inj}_{\rightarrow}(\mathbf{c}_1)), (\mathbf{G}'_2, \text{inj}_{\rightarrow}(\mathbf{c}'_2)))$). We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\text{?}_\delta, (\mathbf{G}'_1, \text{inj}_{\rightarrow}(\mathbf{c}_1)), (\mathbf{G}'_2, \text{inj}_{\rightarrow}(\mathbf{c}'_2))) = \\ &= (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \text{inj}_{\rightarrow}(\mathbf{c}'_5)) \end{aligned}$$

where $\text{pullback}(\text{?}_\delta \rightarrow \text{?}_\delta, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}'_2, \mathbf{c}'_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. By the induction hypothesis we get that

$$\text{pullback}(\text{?}_\delta \rightarrow \text{?}_\delta, (\mathbf{G}'_2, \mathbf{c}'_2), (\mathbf{G}'_1, \mathbf{c}_1)) = (\mathbf{G}'_3, \mathbf{c}'_4, \mathbf{c}'_3, \mathbf{c}'_5)$$

Therefore, we have that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) &= \text{pullback}(\text{?}_\delta, (\mathbf{G}'_2, \text{inj}_{\rightarrow}(\mathbf{c}'_2)), (\mathbf{G}'_1, \text{inj}_{\rightarrow}(\mathbf{c}_1))) = \\ &= (\mathbf{G}'_3, \mathbf{c}'_4, \mathbf{c}'_3, \text{inj}_{\rightarrow}(\mathbf{c}'_5)) \end{aligned}$$

Finally, the results follows immediately.

Case ($\text{pullback}(\delta, (\mathbf{G}'_1, \text{inj}_\times(\mathbf{c}_1)), (\mathbf{G}'_2, \text{inj}_\times(\mathbf{c}'_2)))$). We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\delta, (\mathbf{G}'_1, \text{inj}_\times(\mathbf{c}_1)), (\mathbf{G}'_2, \text{inj}_\times(\mathbf{c}'_2))) = \\ &(\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \text{inj}_\times(\mathbf{c}'_5)) \end{aligned}$$

where $\text{pullback}(\delta \times \delta, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}'_2, \mathbf{c}'_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. By the induction hypothesis we get that

$$\text{pullback}(\delta \times \delta, (\mathbf{G}'_2, \mathbf{c}'_2), (\mathbf{G}'_1, \mathbf{c}_1)) = (\mathbf{G}'_3, \mathbf{c}'_4, \mathbf{c}'_3, \mathbf{c}'_5)$$

Therefore, we have that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) &= \text{pullback}(\delta, (\mathbf{G}'_2, \text{inj}_\times(\mathbf{c}'_2)), (\mathbf{G}'_1, \text{inj}_\times(\mathbf{c}_1))) = \\ &(\mathbf{G}'_3, \mathbf{c}'_4, \mathbf{c}'_3, \text{inj}_\times(\mathbf{c}'_5)) \end{aligned}$$

Finally, the results follows immediately.

Case ($\text{pullback}(\delta, (\mathbf{G}'_1, \text{inj}_\vee(\mathbf{c}_1)), (\mathbf{G}'_2, \text{inj}_\vee(\mathbf{c}'_2)))$). We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\delta, (\mathbf{G}'_1, \text{inj}_\vee(\mathbf{c}_1)), (\mathbf{G}'_2, \text{inj}_\vee(\mathbf{c}'_2))) = \\ &(\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \text{inj}_\vee(\mathbf{c}'_5)) \end{aligned}$$

where $\text{pullback}(\forall \mathbf{Y}.\delta, \mathbf{Y}:\mathbf{Y}, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}'_2, \mathbf{c}'_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. By the induction hypothesis we get that

$$\text{pullback}(\forall \mathbf{Y}.\delta, \mathbf{Y}:\mathbf{Y}, (\mathbf{G}'_2, \mathbf{c}'_2), (\mathbf{G}'_1, \mathbf{c}_1)) = (\mathbf{G}'_3, \mathbf{c}'_4, \mathbf{c}'_3, \mathbf{c}'_5)$$

Therefore, we have that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) &= \text{pullback}(\delta, (\mathbf{G}'_2, \text{inj}_\vee(\mathbf{c}'_2)), (\mathbf{G}'_1, \text{inj}_\vee(\mathbf{c}_1))) = \\ &(\mathbf{G}'_3, \mathbf{c}'_4, \mathbf{c}'_3, \text{inj}_\vee(\mathbf{c}'_5)) \end{aligned}$$

Finally, the results follows immediately.

Case ($\text{pullback}(\delta, (\delta', \text{inj}_\gamma), (\mathbf{G}_2, \mathbf{c}_2))$). We proceed by cases analysis.

- $\mathbf{G}_2 = \mathbf{B}$ and $\mathbf{c}_2 = \text{inj}_\mathbf{B}$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_\gamma), (\mathbf{B}, \text{inj}_\mathbf{B})) = (\mathbf{B}, \text{inj}_\mathbf{B}, \mathbf{B}, \text{inj}_\mathbf{B})$$

Also, we know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = \text{pullback}(\delta, (\mathbf{B}, \text{inj}_\mathbf{B}), (\delta', \text{inj}_\gamma)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$$

where $\text{pullback}(\delta, (\delta', \text{inj}_\gamma), (\mathbf{B}, \text{inj}_\mathbf{B})) = (\mathbf{G}'_3, \mathbf{c}'_4, \mathbf{c}'_3, \mathbf{c}'_5) = (\mathbf{B}, \text{inj}_\mathbf{B}, \mathbf{B}, \text{inj}_\mathbf{B})$. Therefore, $\text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = (\mathbf{B}, \mathbf{B}, \text{inj}_\mathbf{B}, \text{inj}_\mathbf{B})$, as we are required to show.

- $\mathbf{G}_2 = \mathbf{F}$, $\mathbf{c}_2 = \text{inj}_\mathbf{X}$ and $\mathbf{X} : \mathbf{F} \in \delta'$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_\gamma), (\mathbf{F}, \text{inj}_\mathbf{X})) = (\mathbf{F}, \text{inj}_\mathbf{X}, \mathbf{F}, \text{inj}_\mathbf{X})$$

Also, we know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = \text{pullback}(\delta, (\mathbf{F}, \text{inj}_\mathbf{X}), (\delta', \text{inj}_\gamma)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$$

where $\text{pullback}(\delta, (\delta', \text{inj}_\gamma), (\mathbf{F}, \text{inj}_\mathbf{X})) = (\mathbf{G}'_3, \mathbf{c}'_4, \mathbf{c}'_3, \mathbf{c}'_5) = (\mathbf{F}, \text{inj}_\mathbf{X}, \mathbf{F}, \text{inj}_\mathbf{X})$. Therefore, $\text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = (\mathbf{F}, \mathbf{F}, \text{inj}_\mathbf{X}, \text{inj}_\mathbf{X})$, as we are required to show.

- $\mathbf{c}_2 = \text{inj}_{\rightarrow}(\mathbf{c}'_2)$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_{\rightarrow}), (\mathbf{G}_2, \text{inj}_{\rightarrow}(\mathbf{c}'_2))) = (\mathbf{G}'_3, \text{inj}_{\rightarrow}(\mathbf{c}'_3), \mathbf{c}'_4, \text{inj}_{\rightarrow}(\mathbf{c}'_5))$$

where $\text{pullback}(\delta \rightarrow \delta, (\delta' \rightarrow \delta', \text{inj}_{\rightarrow} \rightarrow \text{inj}_{\rightarrow}), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. Also, we know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = \text{pullback}(\delta, (\mathbf{G}_2, \text{inj}_{\rightarrow}(\mathbf{c}'_2)), (\delta', \text{inj}_{\rightarrow})) = (\mathbf{G}''_3, \mathbf{c}''_4, \mathbf{c}''_3, \mathbf{c}''_5)$$

where $\text{pullback}(\delta, (\delta', \text{inj}_{\rightarrow}), (\mathbf{G}_2, \text{inj}_{\rightarrow}(\mathbf{c}'_2))) = (\mathbf{G}''_3, \mathbf{c}''_3, \mathbf{c}''_4, \mathbf{c}''_5) = (\mathbf{G}'_3, \text{inj}_{\rightarrow}(\mathbf{c}'_3), \mathbf{c}'_4, \text{inj}_{\rightarrow}(\mathbf{c}'_5))$. Therefore, $\text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = (\mathbf{G}'_3, \mathbf{c}'_4, \text{inj}_{\rightarrow}(\mathbf{c}'_3), \text{inj}_{\rightarrow}(\mathbf{c}'_5))$, as we are required to show.

- $\mathbf{c}_2 = \text{inj}_{\times}(\mathbf{c}'_2)$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_{\times}), (\mathbf{G}_2, \text{inj}_{\times}(\mathbf{c}'_2))) = (\mathbf{G}'_3, \text{inj}_{\times}(\mathbf{c}'_3), \mathbf{c}'_4, \text{inj}_{\times}(\mathbf{c}'_5))$$

where $\text{pullback}(\delta \times \delta, (\delta' \times \delta', \text{inj}_{\times} \times \text{inj}_{\times}), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. Also, we know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = \text{pullback}(\delta, (\mathbf{G}_2, \text{inj}_{\times}(\mathbf{c}'_2)), (\delta', \text{inj}_{\times})) = (\mathbf{G}''_3, \mathbf{c}''_4, \mathbf{c}''_3, \mathbf{c}''_5)$$

where $\text{pullback}(\delta, (\delta', \text{inj}_{\times}), (\mathbf{G}_2, \text{inj}_{\times}(\mathbf{c}'_2))) = (\mathbf{G}''_3, \mathbf{c}''_3, \mathbf{c}''_4, \mathbf{c}''_5) = (\mathbf{G}'_3, \text{inj}_{\times}(\mathbf{c}'_3), \mathbf{c}'_4, \text{inj}_{\times}(\mathbf{c}'_5))$. Therefore, $\text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = (\mathbf{G}'_3, \mathbf{c}'_4, \text{inj}_{\times}(\mathbf{c}'_3), \text{inj}_{\times}(\mathbf{c}'_5))$, as we are required to show.

- $\mathbf{c}_2 = \text{inj}_{\vee}(\mathbf{c}'_2)$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_{\vee}), (\mathbf{G}_2, \text{inj}_{\vee}(\mathbf{c}'_2))) = (\mathbf{G}'_3, \text{inj}_{\vee}(\mathbf{c}'_3), \mathbf{c}'_4, \text{inj}_{\vee}(\mathbf{c}'_5))$$

where $\text{pullback}(\forall \mathbf{Y}.\delta, \mathbf{Y}:\mathbf{Y}, (\forall \mathbf{Y}.\delta', \mathbf{Y}:\mathbf{Y}, \forall \mathbf{Y}.\text{inj}_{\vee}), (\mathbf{G}_2, \mathbf{c}_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. Also, we know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = \text{pullback}(\delta, (\mathbf{G}_2, \text{inj}_{\vee}(\mathbf{c}'_2)), (\delta', \text{inj}_{\vee})) = (\mathbf{G}''_3, \mathbf{c}''_4, \mathbf{c}''_3, \mathbf{c}''_5)$$

where $\text{pullback}(\delta, (\delta', \text{inj}_{\vee}), (\mathbf{G}_2, \text{inj}_{\vee}(\mathbf{c}'_2))) = (\mathbf{G}''_3, \mathbf{c}''_3, \mathbf{c}''_4, \mathbf{c}''_5) = (\mathbf{G}'_3, \text{inj}_{\vee}(\mathbf{c}'_3), \mathbf{c}'_4, \text{inj}_{\vee}(\mathbf{c}'_5))$. Therefore, $\text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1)) = (\mathbf{G}'_3, \mathbf{c}'_4, \text{inj}_{\vee}(\mathbf{c}'_3), \text{inj}_{\vee}(\mathbf{c}'_5))$, as we are required to show.

- $\mathbf{G}_2 = \delta''$ and $\mathbf{c}_2 = \text{inj}_{\rightarrow}$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_{\rightarrow}), (\delta'', \text{inj}_{\rightarrow})) = (\delta' \cap \delta'', \text{inj}_{\rightarrow}, \text{inj}_{\rightarrow}, \text{inj}_{\rightarrow}) = \text{pullback}(\delta, (\delta', \text{inj}_{\rightarrow}), (\delta'', \text{inj}_{\rightarrow}))$$

Therefore, the result follows immediately.

Case ($\text{pullback}(\delta, (\mathbf{G}_1, \mathbf{c}_1), (\delta', \text{inj}_{\rightarrow}))$). We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\mathbf{G}_1, \mathbf{c}_1), (\delta', \text{inj}_{\rightarrow})) = (\mathbf{G}''_3, \mathbf{c}''_4, \mathbf{c}''_3, \mathbf{c}''_5)$$

where $(\mathbf{G}''_3, \mathbf{c}''_3, \mathbf{c}''_4, \mathbf{c}''_5) = \text{pullback}(\delta, (\delta', \text{inj}_{\rightarrow}), (\mathbf{G}_1, \mathbf{c}_1)) = \text{pullback}(\mathbf{G}, (\mathbf{G}_2, \mathbf{c}_2), (\mathbf{G}_1, \mathbf{c}_1))$. Therefore, the result holds.

The \Leftarrow direction is analogous. □

Case ($\text{pullback}(\forall \mathbf{Y}. \mathbf{G}'_1, (\forall \mathbf{Y}. \mathbf{G}_{11}, \forall \mathbf{Y}. \mathbf{c}_{11}), (\forall \mathbf{Y}. \mathbf{G}_{21}, \forall \mathbf{Y}. \mathbf{c}_{21}))$). We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\forall \mathbf{Y}. \mathbf{G}'_1, (\forall \mathbf{Y}. \mathbf{G}_{11}, \forall \mathbf{Y}. \mathbf{c}_{11}), (\forall \mathbf{Y}. \mathbf{G}_{21}, \forall \mathbf{Y}. \mathbf{c}_{21})) = \\ &(\forall \mathbf{Y}. \mathbf{G}'_3, \forall \mathbf{Y}. \mathbf{c}'_3, \forall \mathbf{Y}. \mathbf{c}'_4, \forall \mathbf{Y}. \mathbf{c}'_5) \end{aligned}$$

where $\text{pullback}(\mathbf{G}'_1, (\mathbf{G}_{11}, \mathbf{c}_{11}), (\mathbf{G}_{21}, \mathbf{c}_{21})) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$ By the induction hypothesis we get that

$$\begin{aligned} \text{pullback}(\mathbf{G}'_1[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}_{11}[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_{11}[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_{21}[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_{21}[\mathbf{X} \mapsto \mathbf{F}])) &= \\ (\mathbf{G}'_3[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}'_3[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}'_4[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}'_5[\mathbf{X} \mapsto \mathbf{F}]) \end{aligned}$$

Therefore, we have that

$$\begin{aligned} \text{pullback}(\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}])) &= \\ (\forall \mathbf{Y}. \mathbf{G}'_3[\mathbf{X} \mapsto \mathbf{F}] \mathbf{G}''_3[\mathbf{X} \mapsto \mathbf{F}], \forall \mathbf{Y}. \mathbf{c}'_3[\mathbf{X} \mapsto \mathbf{F}] \mathbf{c}''_3[\mathbf{X} \mapsto \mathbf{F}], \forall \mathbf{Y}. \mathbf{c}'_4[\mathbf{X} \mapsto \mathbf{F}] \mathbf{c}''_4[\mathbf{X} \mapsto \mathbf{F}], \forall \mathbf{Y}. \mathbf{c}'_5[\mathbf{X} \mapsto \mathbf{F}] \mathbf{c}''_5[\mathbf{X} \mapsto \mathbf{F}]) & \\ (\forall \mathbf{Y}. \mathbf{G}'_3, \forall \mathbf{Y}. \mathbf{c}'_3, \forall \mathbf{Y}. \mathbf{c}'_4, \forall \mathbf{Y}. \mathbf{c}'_5)[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

as we are required to show.

Case ($\text{pullback}(\text{?}_\delta, (\mathbf{B}, \text{inj}_\mathbf{B}), (\mathbf{B}, \text{inj}_\mathbf{B}))$). We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\text{?}_\delta, (\mathbf{B}, \text{inj}_\mathbf{B}), (\mathbf{B}, \text{inj}_\mathbf{B})) = (\mathbf{B}, \mathbf{B}, \mathbf{B}, \text{inj}_\mathbf{B})$$

Thus, we have that

$$\begin{aligned} \text{pullback}(\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}])) &= \\ \text{pullback}(\text{?}_\delta[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{B}[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\mathbf{B}[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{B}[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\mathbf{B}[\mathbf{X} \mapsto \mathbf{F}])) &= \\ \text{pullback}(\text{?}_\delta[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{B}, \text{inj}_\mathbf{B}), (\mathbf{B}, \text{inj}_\mathbf{B})) = (\mathbf{B}, \mathbf{B}, \mathbf{B}, \text{inj}_\mathbf{B}) &= \\ (\mathbf{B}, \mathbf{B}, \mathbf{B}, \text{inj}_\mathbf{B})[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

as we are required to show.

Case ($\text{pullback}(\text{?}_\delta, (\mathbf{F}', \text{inj}_\mathbf{Y}), (\mathbf{F}', \text{inj}_\mathbf{Y}))$). We have the following cases.

- $\mathbf{F}' = \mathbf{X}$. We know that $\mathbf{Y} : \mathbf{X} \in \delta$ and

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\text{?}_\delta, (\mathbf{X}, \text{inj}_\mathbf{Y}), (\mathbf{X}, \text{inj}_\mathbf{Y})) = (\mathbf{X}, \mathbf{X}, \mathbf{X}, \text{inj}_\mathbf{Y})$$

Thus, we have that $\mathbf{Y} : \mathbf{F} \in \delta[\mathbf{X} \mapsto \mathbf{F}]$ and

$$\begin{aligned} \text{pullback}(\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}])) &= \\ \text{pullback}(\text{?}_\delta[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{X}[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\mathbf{Y}[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{X}[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\mathbf{Y}[\mathbf{X} \mapsto \mathbf{F}])) &= \\ \text{pullback}(\text{?}_\delta[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{F}, \text{inj}_\mathbf{Y}), (\mathbf{F}, \text{inj}_\mathbf{Y})) = (\mathbf{F}, \mathbf{F}, \mathbf{F}, \text{inj}_\mathbf{Y}) &= \\ (\mathbf{X}, \mathbf{X}, \mathbf{X}, \text{inj}_\mathbf{Y})[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

as we are required to show.

- $F' \neq X$. We know that $Y : F' \in \delta$ and

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (F', \text{inj}_Y), (F', \text{inj}_Y)) = (F', F', F', \text{inj}_Y)$$

Thus, we have that $Y : F' \in \delta[X \mapsto F]$ and

$$\begin{aligned} & \text{pullback}(\mathbf{G}[X \mapsto F], (\mathbf{G}_1[X \mapsto F], \mathbf{c}_1[X \mapsto F]), (\mathbf{G}_2[X \mapsto F], \mathbf{c}_2[X \mapsto F])) = \\ & \text{pullback}(\delta[X \mapsto F], (F'[X \mapsto F], \text{inj}_Y[X \mapsto F]), (F'[X \mapsto F], \text{inj}_Y[X \mapsto F])) = \\ & \text{pullback}(\delta[X \mapsto F], (F', \text{inj}_Y), (F', \text{inj}_Y)) = (F', F', F', \text{inj}_Y) = \\ & (F', F', F', \text{inj}_Y)[X \mapsto F] \end{aligned}$$

as we are required to show.

Case $(\text{pullback}(\delta, (\mathbf{G}'_1, \text{inj}_{\rightarrow}(\mathbf{c}_1)), (\mathbf{G}'_2, \text{inj}_{\rightarrow}(\mathbf{c}'_2))))$. We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\delta, (\mathbf{G}'_1, \text{inj}_{\rightarrow}(\mathbf{c}_1)), (\mathbf{G}'_2, \text{inj}_{\rightarrow}(\mathbf{c}'_2))) = \\ & (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \text{inj}_{\rightarrow}(\mathbf{c}'_5)) \end{aligned}$$

where $\text{pullback}(\delta \rightarrow \delta, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}'_2, \mathbf{c}'_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. By the induction hypothesis we get that

$$\text{pullback}(\delta[X \mapsto F] \rightarrow \delta[X \mapsto F], (\mathbf{G}'_1[X \mapsto F], \mathbf{c}_1[X \mapsto F]), (\mathbf{G}'_2[X \mapsto F], \mathbf{c}'_2[X \mapsto F])) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)[X \mapsto F]$$

Therefore, we have that

$$\begin{aligned} & \text{pullback}(\mathbf{G}[X \mapsto F], (\mathbf{G}_1[X \mapsto F], \mathbf{c}_1[X \mapsto F]), (\mathbf{G}_2[X \mapsto F], \mathbf{c}_2[X \mapsto F])) = \\ & \text{pullback}(\delta[X \mapsto F], (\mathbf{G}'_1[X \mapsto F], \text{inj}_{\rightarrow}(\mathbf{c}_1[X \mapsto F])), (\mathbf{G}'_2[X \mapsto F], \text{inj}_{\rightarrow}(\mathbf{c}'_2[X \mapsto F]))) = \\ & (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \text{inj}_{\rightarrow}(\mathbf{c}'_5))[X \mapsto F] \end{aligned}$$

as we are required to show.

Case $(\text{pullback}(\delta, (\mathbf{G}'_1, \text{inj}_{\times}(\mathbf{c}_1)), (\mathbf{G}'_2, \text{inj}_{\times}(\mathbf{c}'_2))))$. We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\delta, (\mathbf{G}'_1, \text{inj}_{\times}(\mathbf{c}_1)), (\mathbf{G}'_2, \text{inj}_{\times}(\mathbf{c}'_2))) = \\ & (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \text{inj}_{\times}(\mathbf{c}'_5)) \end{aligned}$$

where $\text{pullback}(\delta \times \delta, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}'_2, \mathbf{c}'_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. By the induction hypothesis we get that

$$\text{pullback}(\delta[X \mapsto F] \times \delta[X \mapsto F], (\mathbf{G}'_1[X \mapsto F], \mathbf{c}_1[X \mapsto F]), (\mathbf{G}'_2[X \mapsto F], \mathbf{c}'_2[X \mapsto F])) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)[X \mapsto F]$$

Therefore, we have that

$$\begin{aligned} & \text{pullback}(\mathbf{G}[X \mapsto F], (\mathbf{G}_1[X \mapsto F], \mathbf{c}_1[X \mapsto F]), (\mathbf{G}_2[X \mapsto F], \mathbf{c}_2[X \mapsto F])) = \\ & \text{pullback}(\delta[X \mapsto F], (\mathbf{G}'_1[X \mapsto F], \text{inj}_{\times}(\mathbf{c}_1[X \mapsto F])), (\mathbf{G}'_2[X \mapsto F], \text{inj}_{\times}(\mathbf{c}'_2[X \mapsto F]))) = \\ & (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \text{inj}_{\times}(\mathbf{c}'_5))[X \mapsto F] \end{aligned}$$

as we are required to show.

Case ($\text{pullback}(\delta, (\mathbf{G}'_1, \text{inj}_\forall(\mathbf{c}_1)), (\mathbf{G}'_2, \text{inj}_\forall(\mathbf{c}'_2)))$). We know that

$$\begin{aligned} \text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) &= \text{pullback}(\delta, (\mathbf{G}'_1, \text{inj}_\forall(\mathbf{c}_1)), (\mathbf{G}'_2, \text{inj}_\forall(\mathbf{c}'_2))) = \\ &(\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \text{inj}_\forall(\mathbf{c}'_5)) \end{aligned}$$

where $\text{pullback}(\forall \mathbf{Y}.\delta, \mathbf{Y}:\mathbf{Y}, (\mathbf{G}'_1, \mathbf{c}_1), (\mathbf{G}'_2, \mathbf{c}'_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. By the induction hypothesis we get that

$$\text{pullback}(\forall \mathbf{Y}.\delta, \mathbf{Y}:\mathbf{Y})[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}'_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}'_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}])) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)[\mathbf{X} \mapsto \mathbf{F}]$$

Therefore, we have that

$$\begin{aligned} \text{pullback}(\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}])) &= \\ \text{pullback}(\delta[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}'_1[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\forall(\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}])), (\mathbf{G}'_2[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\forall(\mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}]))) &= \\ (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \text{inj}_\forall(\mathbf{c}'_5))[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

as we are required to show.

Case ($\text{pullback}(\delta, (\delta', \text{inj}_\gamma), (\mathbf{G}_2, \mathbf{c}_2))$). We proceed by cases analysis.

- $\mathbf{G}_2 = \mathbf{B}$ and $\mathbf{c}_2 = \text{inj}_\mathbf{B}$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_\gamma), (\mathbf{B}, \text{inj}_\mathbf{B})) = (\mathbf{B}, \text{inj}_\mathbf{B}, \mathbf{B}, \text{inj}_\mathbf{B})$$

Also, we know that

$$\begin{aligned} \text{pullback}(\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}])) &= \\ \text{pullback}(\delta[\mathbf{X} \mapsto \mathbf{F}], (\delta'[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\gamma[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{B}[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\mathbf{B}[\mathbf{X} \mapsto \mathbf{F}])) &= \\ \text{pullback}(\delta[\mathbf{X} \mapsto \mathbf{F}], (\delta'[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\gamma), (\mathbf{B}, \text{inj}_\mathbf{B})) &= \\ (\mathbf{B}, \text{inj}_\mathbf{B}, \mathbf{B}, \text{inj}_\mathbf{B}) = (\mathbf{B}, \text{inj}_\mathbf{B}, \mathbf{B}, \text{inj}_\mathbf{B})[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

as we are required to show.

- $\mathbf{G}_2 = \mathbf{F}'$, $\mathbf{c}_2 = \text{inj}_\mathbf{Y}$ and $\mathbf{Y} : \mathbf{F}' \in \delta'$.

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_\gamma), (\mathbf{F}', \text{inj}_\mathbf{Y})) = (\mathbf{F}', \text{inj}_\mathbf{Y}, \mathbf{F}', \text{inj}_\mathbf{Y})$$

We have the following cases.

- $\mathbf{F}' = \mathbf{X}$. We know that $\mathbf{Y} : \mathbf{X} \in \delta'$ and

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_\gamma), (\mathbf{X}, \text{inj}_\mathbf{Y})) = (\mathbf{X}, \text{inj}_\mathbf{Y}, \mathbf{X}, \text{inj}_\mathbf{Y})$$

Thus, we have that $\mathbf{Y} : \mathbf{F} \in \delta'[\mathbf{X} \mapsto \mathbf{F}]$ and

$$\begin{aligned} \text{pullback}(\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}])) &= \\ \text{pullback}(\delta[\mathbf{X} \mapsto \mathbf{F}], (\delta'[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\gamma[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{X}[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\mathbf{Y}[\mathbf{X} \mapsto \mathbf{F}])) &= \\ \text{pullback}(\delta[\mathbf{X} \mapsto \mathbf{F}], (\delta'[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\gamma), (\mathbf{F}, \text{inj}_\mathbf{Y})) = (\mathbf{F}, \text{inj}_\mathbf{Y}, \mathbf{F}, \text{inj}_\mathbf{Y}) &= \\ (\mathbf{X}, \text{inj}_\mathbf{Y}, \mathbf{X}, \text{inj}_\mathbf{Y})[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

as we are required to show.

– $\mathbf{F}' \neq \mathbf{X}$. We know that $\mathbf{Y} : \mathbf{F}' \in \delta'$ and

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_?), (\mathbf{F}', \text{inj}_{\mathbf{Y}})) = (\mathbf{F}', \text{inj}_{\mathbf{Y}}, \mathbf{F}', \text{inj}_{\mathbf{Y}})$$

Thus, we have that $\mathbf{Y} : \mathbf{F}' \in \delta'[\mathbf{X} \mapsto \mathbf{F}]$ and

$$\begin{aligned} & \text{pullback}(\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}])) = \\ & \text{pullback}(\delta[\mathbf{X} \mapsto \mathbf{F}], (\delta'[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_?[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{F}'[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_{\mathbf{Y}}[\mathbf{X} \mapsto \mathbf{F}])) = \\ & \text{pullback}(\delta[\mathbf{X} \mapsto \mathbf{F}], (\delta'[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_?), (\mathbf{F}', \text{inj}_{\mathbf{Y}})) = (\mathbf{F}', \text{inj}_{\mathbf{Y}}, \mathbf{F}', \text{inj}_{\mathbf{Y}}) = \\ & (\mathbf{F}', \text{inj}_{\mathbf{Y}}, \mathbf{F}', \text{inj}_{\mathbf{Y}})[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

as we are required to show.

- $\mathbf{c}_2 = \text{inj}_{\rightarrow}(\mathbf{c}'_2)$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_?), (\mathbf{G}_2, \text{inj}_{\rightarrow}(\mathbf{c}'_2))) = (\mathbf{G}'_3, \text{inj}_{\rightarrow}(\mathbf{c}'_3), \mathbf{c}'_4, \text{inj}_{\rightarrow}(\mathbf{c}'_5))$$

where $\text{pullback}(\delta \rightarrow \delta, (\delta' \rightarrow \delta', \text{inj}_? \rightarrow \text{inj}_?), (\mathbf{G}_2, \mathbf{c}'_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. By the induction hypothesis, we have that

$$\begin{aligned} & \text{pullback}((\delta \rightarrow \delta)[\mathbf{X} \mapsto \mathbf{F}], ((\delta' \rightarrow \delta')[\mathbf{X} \mapsto \mathbf{F}], (\text{inj}_? \rightarrow \text{inj}_?)[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}])) = \\ & (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

Also, we know that

$$\begin{aligned} & \text{pullback}(\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}])) = \\ & \text{pullback}(\delta[\mathbf{X} \mapsto \mathbf{F}], (\delta'[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_?[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_{\rightarrow}(\mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}]))) = \\ & (\mathbf{G}'_3[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_{\rightarrow}(\mathbf{c}'_3[\mathbf{X} \mapsto \mathbf{F}]), \mathbf{c}'_4[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_{\rightarrow}(\mathbf{c}'_5[\mathbf{X} \mapsto \mathbf{F}])) = \\ & (\mathbf{G}'_3, \text{inj}_{\rightarrow}(\mathbf{c}'_3), \mathbf{c}'_4, \text{inj}_{\rightarrow}(\mathbf{c}'_5))[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

as we are required to show.

- $\mathbf{c}_2 = \text{inj}_{\times}(\mathbf{c}'_2)$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_?), (\mathbf{G}_2, \text{inj}_{\times}(\mathbf{c}'_2))) = (\mathbf{G}'_3, \text{inj}_{\times}(\mathbf{c}'_3), \mathbf{c}'_4, \text{inj}_{\times}(\mathbf{c}'_5))$$

where $\text{pullback}(\delta \times \delta, (\delta' \times \delta', \text{inj}_? \times \text{inj}_?), (\mathbf{G}_2, \mathbf{c}'_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. By the induction hypothesis, we have that

$$\begin{aligned} & \text{pullback}((\delta \times \delta)[\mathbf{X} \mapsto \mathbf{F}], ((\delta' \times \delta')[\mathbf{X} \mapsto \mathbf{F}], (\text{inj}_? \times \text{inj}_?)[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}])) = \\ & (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

Also, we know that

$$\begin{aligned} & \text{pullback}(\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}])) = \\ & \text{pullback}(\delta[\mathbf{X} \mapsto \mathbf{F}], (\delta'[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_?[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_{\times}(\mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}]))) = \\ & (\mathbf{G}'_3[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_{\times}(\mathbf{c}'_3[\mathbf{X} \mapsto \mathbf{F}]), \mathbf{c}'_4[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_{\times}(\mathbf{c}'_5[\mathbf{X} \mapsto \mathbf{F}])) = \\ & (\mathbf{G}'_3, \text{inj}_{\times}(\mathbf{c}'_3), \mathbf{c}'_4, \text{inj}_{\times}(\mathbf{c}'_5))[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

as we are required to show.

- $\mathbf{c}_2 = \text{inj}_\forall(\mathbf{c}'_2)$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_?), (\mathbf{G}_2, \text{inj}_\forall(\mathbf{c}'_2))) = (\mathbf{G}'_3, \text{inj}_\forall(\mathbf{c}'_3), \mathbf{c}'_4, \text{inj}_\forall(\mathbf{c}'_5))$$

where $\text{pullback}(\forall \mathbf{Y}.\delta, \mathbf{Y}:\mathbf{Y}, (\forall \mathbf{Y}.\delta', \mathbf{Y}:\mathbf{Y}, \forall \mathbf{Y}.\text{inj}_?), (\mathbf{G}_2, \mathbf{c}'_2)) = (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)$. By the induction hypothesis, we have that

$$\begin{aligned} \text{pullback}((\forall \mathbf{Y}.\delta, \mathbf{Y}:\mathbf{Y})[\mathbf{X} \mapsto \mathbf{F}], ((\forall \mathbf{Y}.\delta', \mathbf{Y}:\mathbf{Y})[\mathbf{X} \mapsto \mathbf{F}], (\forall \mathbf{Y}.\text{inj}_?)[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}])) = \\ (\mathbf{G}'_3, \mathbf{c}'_3, \mathbf{c}'_4, \mathbf{c}'_5)[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

Also, we know that

$$\begin{aligned} \text{pullback}(\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}])) = \\ \text{pullback}(\delta[\mathbf{X} \mapsto \mathbf{F}], (\delta'[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_?[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\forall(\mathbf{c}'_2[\mathbf{X} \mapsto \mathbf{F}])) = \\ (\mathbf{G}'_3[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\forall(\mathbf{c}'_3[\mathbf{X} \mapsto \mathbf{F}]), \mathbf{c}'_4[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_\forall(\mathbf{c}'_5[\mathbf{X} \mapsto \mathbf{F}])) = \\ (\mathbf{G}'_3, \text{inj}_\forall(\mathbf{c}'_3), \mathbf{c}'_4, \text{inj}_\forall(\mathbf{c}'_5))[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

as we are required to show.

- $\mathbf{G}_2 = \delta''$ and $\mathbf{c}_2 = \text{inj}_?$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\delta', \text{inj}_?), (\delta'', \text{inj}_?)) = (\delta' \cap \delta'', \text{inj}_?, \text{inj}_?, \text{inj}_?)$$

Also, we know that

$$\begin{aligned} \text{pullback}(\mathbf{G}[\mathbf{X} \mapsto \mathbf{F}], (\mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]), (\mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}])) = \\ \text{pullback}(\delta[\mathbf{X} \mapsto \mathbf{F}], (\delta'[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_?[\mathbf{X} \mapsto \mathbf{F}]), (\delta''[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_?[\mathbf{X} \mapsto \mathbf{F}])) = \\ (\delta' \cap \delta''[\mathbf{X} \mapsto \mathbf{F}], \text{inj}_?, \text{inj}_?, \text{inj}_?) = \\ (\delta' \cap \delta'', \text{inj}_?, \text{inj}_?, \text{inj}_?)[\mathbf{X} \mapsto \mathbf{F}] \end{aligned}$$

as we are required to show.

Case $(\text{pullback}(\delta, (\mathbf{G}_1, \mathbf{c}_1), (\delta', \text{inj}_?)))$. We know that

$$\text{pullback}(\mathbf{G}, (\mathbf{G}_1, \mathbf{c}_1), (\mathbf{G}_2, \mathbf{c}_2)) = \text{pullback}(\delta, (\mathbf{G}_1, \mathbf{c}_1), (\delta', \text{inj}_?)) = (\mathbf{G}''_3, \mathbf{c}''_4, \mathbf{c}''_3, \mathbf{c}''_5)$$

where

$$\begin{aligned} (\mathbf{G}''_3, \mathbf{c}''_3, \mathbf{c}''_4, \mathbf{c}''_5) = \text{pullback}(\delta, (\delta', \text{inj}_?), (\mathbf{G}_1, \mathbf{c}_1)) = \\ \text{pullback}(\mathbf{G}, (\delta', \text{inj}_?), (\mathbf{G}_1, \mathbf{c}_1)) \end{aligned}$$

Thus, this case can be resolved with the cases above.

The \Leftarrow direction is analogous. □

We use the following definition of $\rho_i(\mathbf{t})$:

$$\begin{aligned} (\rho, \mathbf{X} \mapsto (\mathbf{F}_1, \mathbf{F}_2, R))_i(\mathbf{t}) &= \rho_i(\mathbf{t}[\mathbf{F}_i/\mathbf{X}]) \\ \cdot_i(\mathbf{t}) &= \mathbf{t} \end{aligned}$$

We use notation

$$\begin{aligned} (\rho, \mathbf{X} \mapsto \mathbf{G}_1)(\delta) &= \rho([\mathbf{G}_1/\mathbf{X}]\delta) \\ (\cdot)(\delta) &= \delta \end{aligned}$$

$$c(\mathbf{G}) = \begin{cases} \mathbf{B} & \mathbf{G} = \mathbf{B} \\ \mathbf{X} & \mathbf{G} = \mathbf{X} \\ \rightarrow & \mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2 \\ \times & \mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2 \\ \forall & \mathbf{G} = \forall \mathbf{X}. \mathbf{G}' \end{cases} \quad \mathcal{S}(\delta, \mathbf{G}) = \begin{cases} \mathbf{B} & \mathbf{G} = \mathbf{B} \\ \mathbf{X} & \mathbf{G} = \mathbf{X} \\ \delta \rightarrow \delta & \mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2 \\ \delta \times \delta & \mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2 \\ \forall \mathbf{X}. \mathbf{X}. \delta & \mathbf{G} = \forall \mathbf{X}. \mathbf{G}' \end{cases}$$

Definition B.69 Throughout the proofs we apply induction on this relation. We say that

$$\begin{aligned} (n, \mathbf{G}) < (n', \mathbf{G}') &\iff n < n' \vee (n = n' \wedge \mathbf{G} < \mathbf{G}') \\ (n, \mathbf{G}_1, \mathbf{G}_2) < (n', \mathbf{G}'_1, \mathbf{G}'_2) &\iff n < n' \vee (n = n' \wedge (\mathbf{G}_1, \mathbf{G}_2) < (\mathbf{G}'_1, \mathbf{G}'_2)) \\ (\mathbf{G}_1, \mathbf{G}_2) < (\mathbf{G}'_1, \mathbf{G}'_2) &\iff (\mathbf{G}_1 < \mathbf{G}'_1 \wedge \mathbf{G}_2 < \mathbf{G}'_2) \vee \\ &\quad (\mathbf{G}_1 = \mathbf{G}'_1 \wedge \mathbf{G}_2 < \mathbf{G}'_2) \vee \\ &\quad (\mathbf{G}_1 < \mathbf{G}'_1 \wedge \mathbf{G}_2 = \mathbf{G}'_2) \end{aligned}$$

$$\begin{array}{c} \frac{\delta' \subset \delta}{\delta' < \delta} \quad \frac{}{\mathbf{B} < \delta} \quad \frac{\mathbf{X} : \mathbf{F} \in \delta}{\mathbf{X} < \delta} \quad \frac{\mathbf{G}_1 \leq \delta \quad \mathbf{G}_2 \leq \delta}{\mathbf{G}_1 \rightarrow \mathbf{G}_2 < \delta} \quad \frac{\mathbf{G} \leq \delta, \mathbf{X} : \mathbf{X}}{\forall \mathbf{X}. \mathbf{G} < \delta} \\ \\ \frac{\mathbf{G}_1 \leq \delta \quad \mathbf{G}_2 \leq \delta}{\mathbf{G}_1 \times \mathbf{G}_2 < \delta} \end{array}$$

Lemma B.70 $\{\mathbf{S}_1\} : \mathbf{G}_1 \sim \mathbf{G}$, $\{\mathbf{S}_2\} : \mathbf{G} \sim \mathbf{G}_2$ and $(n, \rho) \in \mathcal{D}[\Delta]$, then $\rho_i(\{\mathbf{S}_1\} \circ \{\mathbf{S}_2\}) = \rho_i(\{\mathbf{S}_1\}) \circ \rho_i(\{\mathbf{S}_2\})$.

Lemma B.71 If $\rho \subseteq \rho'$, then $\text{ATOM}_\rho[\mathbf{G}] \subseteq \text{ATOM}_{\rho'}[\mathbf{G}]$.

Lemma B.72 Let $\Delta \vdash \mathbf{G}$, $\rho' = (\rho; \mathbf{X} \mapsto (\mathbf{B}_1, \mathbf{B}_2, R))$ and $(n, \rho') \in \mathcal{D}[\Delta, \mathbf{X}]$. Then $\text{ATOM}_\rho[\mathbf{G}] = \text{ATOM}_{\rho'}[\mathbf{G}]$.

B.4 $F_\varepsilon^?$: Gradual Guarantees

$\Omega \vdash c : s \sqsubseteq s$ Term precision

$$\begin{array}{c}
\frac{}{\sqsubseteq_b \Omega \vdash \text{refl}_B : b \sqsubseteq b} \qquad \frac{}{\sqsubseteq_x \Omega \vdash c : x \sqsubseteq x} \\
\frac{}{\sqsubseteq_\lambda \Omega, x \mapsto c : G'_1 \sqsubseteq G'_2 \vdash c' : t_1 \sqsubseteq t_2} \qquad \frac{}{\sqsubseteq_\Lambda \Omega \vdash c : t_1 \sqsubseteq t_2} \\
\frac{}{\sqsubseteq_{\times} \Omega \vdash c : s_1 \sqsubseteq s_2 \quad \Omega \vdash c' : s'_1 \sqsubseteq s'_2} \qquad \frac{}{\sqsubseteq_{\text{pairi}} \Omega \vdash c_1 \times c_2 : t_1 \sqsubseteq t_2} \\
\frac{}{\sqsubseteq_{\text{app}} \Omega \vdash c' \longrightarrow c : t_1 \sqsubseteq t_2 \quad \Omega \vdash c' : t'_1 \sqsubseteq t'_2} \qquad \frac{}{\sqsubseteq_{\text{appG}} \Omega \vdash \forall X.c : t_1 \sqsubseteq t_2} \\
\frac{}{\sqsubseteq_{\text{asc}} \Omega \vdash c' : s_1 \sqsubseteq s_2 \quad c : G_1 \sqsubseteq G_2 \quad [c']_{\varepsilon_1} \sqsubseteq \varepsilon_2[c]} \qquad \frac{}{\Omega \vdash c : \varepsilon_1 s_1 :: G_1 \sqsubseteq \varepsilon_2 s_2 :: G_2}
\end{array}$$

Figure B.2: $F_g^?$: Term Precision

Definition B.73 Let $c : G_1 \sqsubseteq G_2$, then $c[X \mapsto \text{refl}_F]$ is defined as:

$$\begin{aligned}
B[X \mapsto \text{refl}_F] &= B \\
X[X \mapsto \text{refl}_F] &= F \\
Y[X \mapsto \text{refl}_F] &= Y \\
(c_1 \longrightarrow c_2)[X \mapsto \text{refl}_F] &= c_1[X \mapsto \text{refl}_F] \longrightarrow c_2[X \mapsto \text{refl}_F] \\
(c_1 \times c_2)[X \mapsto \text{refl}_F] &= c_1[X \mapsto \text{refl}_F] \times c_2[X \mapsto \text{refl}_F] \\
(\forall Y.c_1)[X \mapsto \text{refl}_F] &= \forall Y.(c_1[X \mapsto \text{refl}_F]) \\
\text{inj}_B[X \mapsto \text{refl}_F] &= \text{inj}_B \\
\text{inj}_X[X \mapsto \text{refl}_F] &= \text{inj}_X \\
\text{inj}_{\rightarrow}(c_1)[X \mapsto \text{refl}_F] &= \text{inj}_{\rightarrow}(c_1[X \mapsto \text{refl}_F]) \\
\text{inj}_{\times}(c_1)[X \mapsto \text{refl}_F] &= \text{inj}_{\times}(c_1[X \mapsto \text{refl}_F]) \\
\text{inj}_{\vee}(c_1)[X \mapsto \text{refl}_F] &= \text{inj}_{\vee}(c_1[X \mapsto \text{refl}_F]) \\
\text{inj}_{?}[X \mapsto \text{refl}_F] &= \text{inj}_{?}
\end{aligned}$$

Lemma B.74 $[c_1]_{\varepsilon_1} \sqsubseteq \varepsilon_2[c_2]$ and $[c'_1]_{\varepsilon'_1} \sqsubseteq \varepsilon'_2[c'_2]$ if and only if $[c_1 \longrightarrow c'_1](\varepsilon_1 \longrightarrow \varepsilon'_1) \sqsubseteq (\varepsilon_2 \longrightarrow \varepsilon'_2)[c_2 \longrightarrow c'_2]$.

PROOF. We are required to prove that for all $(G_1 \longrightarrow G'_1, c_{11} \longrightarrow c'_{11}, c_{12} \longrightarrow c'_{12}) \in (\varepsilon_1 \longrightarrow \varepsilon'_1)$ there exists $(G_2 \longrightarrow G'_2, c_{21} \longrightarrow c'_{21}, c_{22} \longrightarrow c'_{22}) \in (\varepsilon_2 \longrightarrow \varepsilon'_2)$, c_l , c_r and c_{tt} such that

- $c_{tt} : G_1 \longrightarrow G'_1 \sqsubseteq G_2 \longrightarrow G'_2$
- $c_{tt}; c_{21} \longrightarrow c'_{21} = c_l$
- $c_{11} \longrightarrow c'_{11}; c_l \longrightarrow c_l = c_r$

- $\mathbf{c}_{tt};\mathbf{c}_{22} \longrightarrow \mathbf{c}'_{22} = \mathbf{c}_r$
- $\mathbf{c}_{12} \longrightarrow \mathbf{c}'_{12};\mathbf{c}_2 \longrightarrow \mathbf{c}'_2 = \mathbf{c}_r$

We know by the definition of $(\varepsilon_1 \longrightarrow \varepsilon'_1)$ that

- $(\mathbf{G}_1, \mathbf{c}_{11}, \mathbf{c}_{12}) \in \varepsilon_1$
- $(\mathbf{G}'_1, \mathbf{c}'_{11}, \mathbf{c}'_{12}) \in \varepsilon'_1$

Also, we know by $[\mathbf{c}_1]\varepsilon_1 \sqsubseteq \varepsilon_2[\mathbf{c}_2]$ and $[\mathbf{c}_1]\varepsilon'_1 \sqsubseteq \varepsilon'_2[\mathbf{c}'_2]$, that for $(\mathbf{G}_1, \mathbf{c}_{11}, \mathbf{c}_{12}) \in \varepsilon_1$ and $(\mathbf{G}'_1, \mathbf{c}'_{11}, \mathbf{c}'_{12}) \in \varepsilon'_1$, there exist $(\mathbf{G}_2, \mathbf{c}_{21}, \mathbf{c}_{22}) \in \varepsilon_2$, $\mathbf{c}''_1, \mathbf{c}''_2, \mathbf{c}_t$, and $(\mathbf{G}'_2, \mathbf{c}'_{21}, \mathbf{c}'_{22}) \in \varepsilon'_2$, $\mathbf{c}'''_1, \mathbf{c}'''_2$ and \mathbf{c}'_t , such that

- $\mathbf{c}_t : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$
- $\mathbf{c}_t;\mathbf{c}_{21} = \mathbf{c}''_1$
- $\mathbf{c}_{11};\mathbf{c}_1 = \mathbf{c}''_1$
- $\mathbf{c}_t;\mathbf{c}_{22} = \mathbf{c}''_2$
- $\mathbf{c}_{12};\mathbf{c}_2 = \mathbf{c}''_2$
- $\mathbf{c}'_t : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$
- $\mathbf{c}'_t;\mathbf{c}'_{21} = \mathbf{c}'''_1$
- $\mathbf{c}'_{11};\mathbf{c}'_1 = \mathbf{c}'''_1$
- $\mathbf{c}'_t;\mathbf{c}'_{22} = \mathbf{c}'''_2$
- $\mathbf{c}'_{12};\mathbf{c}'_2 = \mathbf{c}'''_2$

Therefore, we know that

- $\mathbf{c}_t \longrightarrow \mathbf{c}'_t : \mathbf{G}_1 \longrightarrow \mathbf{G}'_1 \sqsubseteq \mathbf{G}_2 \longrightarrow \mathbf{G}'_2$
- $\mathbf{c}_t \longrightarrow \mathbf{c}'_t;\mathbf{c}_{21} \longrightarrow \mathbf{c}'_{21} = \mathbf{c}''_1 \longrightarrow \mathbf{c}'''_1$
- $\mathbf{c}_{11} \longrightarrow \mathbf{c}'_{11};\mathbf{c}_1 \longrightarrow \mathbf{c}_1 = \mathbf{c}''_1 \longrightarrow \mathbf{c}'''_1$
- $\mathbf{c}_t \longrightarrow \mathbf{c}'_t;\mathbf{c}_{22} \longrightarrow \mathbf{c}'_{22} = \mathbf{c}''_2 \longrightarrow \mathbf{c}'''_2$
- $\mathbf{c}_{12} \longrightarrow \mathbf{c}'_{12};\mathbf{c}_2 \longrightarrow \mathbf{c}'_2 = \mathbf{c}''_2 \longrightarrow \mathbf{c}'''_2$

Thus, taking $\mathbf{c}_1 = \mathbf{c}''_1 \longrightarrow \mathbf{c}'''_1$, $\mathbf{c}_r = \mathbf{c}''_2 \longrightarrow \mathbf{c}'''_2$ and $\mathbf{c}_{tt} = \mathbf{c}_t \longrightarrow \mathbf{c}'_t$, the result follows immediately. \square

Lemma B.75 $[\mathbf{c}_1]\varepsilon_1 \sqsubseteq \varepsilon_2[\mathbf{c}_2]$ and $[\mathbf{c}'_1]\varepsilon'_1 \sqsubseteq \varepsilon'_2[\mathbf{c}'_2]$ if and only if $[\mathbf{c}_1 \times \mathbf{c}'_1](\varepsilon_1 \times \varepsilon'_1) \sqsubseteq (\varepsilon_2 \times \varepsilon'_2)[\mathbf{c}_2 \times \mathbf{c}'_2]$.

PROOF. We are required to prove that for all

$(\mathbf{G}_1 \times \mathbf{G}'_1, \mathbf{c}_{11} \times \mathbf{c}'_{11}, \mathbf{c}_{12} \times \mathbf{c}'_{12}) \in (\varepsilon_1 \times \varepsilon'_1)$ there exists $(\mathbf{G}_2 \times \mathbf{G}'_2, \mathbf{c}_{21} \times \mathbf{c}'_{21}, \mathbf{c}_{22} \times \mathbf{c}'_{22}) \in (\varepsilon_2 \times \varepsilon'_2)$, \mathbf{c}_1 , \mathbf{c}_r and \mathbf{c}_{tt} such that

- $\mathbf{c}_{tt} : \mathbf{G}_1 \times \mathbf{G}'_1 \sqsubseteq \mathbf{G}_2 \times \mathbf{G}'_2$
- $\mathbf{c}_{tt}; \mathbf{c}_{21} \times \mathbf{c}'_{21} = \mathbf{c}_1$
- $\mathbf{c}_{11} \times \mathbf{c}'_{11}; \mathbf{c}_1 \times \mathbf{c}_1 = \mathbf{c}_1$
- $\mathbf{c}_{tt}; \mathbf{c}_{22} \times \mathbf{c}'_{22} = \mathbf{c}_r$
- $\mathbf{c}_{12} \times \mathbf{c}'_{12}; \mathbf{c}_2 \times \mathbf{c}'_2 = \mathbf{c}_r$

We know by the definition of $(\varepsilon_1 \times \varepsilon'_1)$ that

- $(\mathbf{G}_1, \mathbf{c}_{11}, \mathbf{c}_{12}) \in \varepsilon_1$
- $(\mathbf{G}'_1, \mathbf{c}'_{11}, \mathbf{c}'_{12}) \in \varepsilon'_1$

Also, we know by $[\mathbf{c}_1]_{\varepsilon_1} \sqsubseteq \varepsilon_2[\mathbf{c}_2]$ and $[\mathbf{c}_1]_{\varepsilon'_1} \sqsubseteq \varepsilon'_2[\mathbf{c}'_2]$, that for $(\mathbf{G}_1, \mathbf{c}_{11}, \mathbf{c}_{12}) \in \varepsilon_1$ and $(\mathbf{G}'_1, \mathbf{c}'_{11}, \mathbf{c}'_{12}) \in \varepsilon'_1$, there exist $(\mathbf{G}_2, \mathbf{c}_{21}, \mathbf{c}_{22}) \in \varepsilon_2$, \mathbf{c}''_1 , \mathbf{c}''_2 , \mathbf{c}_t , and $(\mathbf{G}'_2, \mathbf{c}'_{21}, \mathbf{c}'_{22}) \in \varepsilon'_2$, \mathbf{c}'''_1 , \mathbf{c}'''_2 and \mathbf{c}'_t , such that

- $\mathbf{c}_t : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$
- $\mathbf{c}_t; \mathbf{c}_{21} = \mathbf{c}''_1$
- $\mathbf{c}_{11}; \mathbf{c}_1 = \mathbf{c}''_1$
- $\mathbf{c}_t; \mathbf{c}_{22} = \mathbf{c}''_2$
- $\mathbf{c}_{12}; \mathbf{c}_2 = \mathbf{c}''_2$
- $\mathbf{c}'_t : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$
- $\mathbf{c}'_t; \mathbf{c}'_{21} = \mathbf{c}'''_1$
- $\mathbf{c}'_{11}; \mathbf{c}'_1 = \mathbf{c}'''_1$
- $\mathbf{c}'_t; \mathbf{c}'_{22} = \mathbf{c}'''_2$
- $\mathbf{c}'_{12}; \mathbf{c}'_2 = \mathbf{c}'''_2$

Therefore, we know that

- $\mathbf{c}_t \times \mathbf{c}'_t : \mathbf{G}_1 \times \mathbf{G}'_1 \sqsubseteq \mathbf{G}_2 \times \mathbf{G}'_2$
- $\mathbf{c}_t \times \mathbf{c}'_t; \mathbf{c}_{21} \times \mathbf{c}'_{21} = \mathbf{c}''_1 \times \mathbf{c}'''_1$
- $\mathbf{c}_{11} \times \mathbf{c}'_{11}; \mathbf{c}_1 \times \mathbf{c}'_1 = \mathbf{c}''_1 \times \mathbf{c}'''_1$
- $\mathbf{c}_t \times \mathbf{c}'_t; \mathbf{c}_{22} \times \mathbf{c}'_{22} = \mathbf{c}''_2 \times \mathbf{c}'''_2$

- $\mathbf{c}_{12} \times \mathbf{c}'_{12}; \mathbf{c}_2 \times \mathbf{c}'_2 = \mathbf{c}''_2 \times \mathbf{c}'''_2$

Thus, taking $\mathbf{c}_1 = \mathbf{c}''_1 \times \mathbf{c}'''_1$, $\mathbf{c}_r = \mathbf{c}''_2 \times \mathbf{c}'''_2$ and $\mathbf{c}_{tt} = \mathbf{c}_t \times \mathbf{c}'_t$, the result follows immediately. \square

Lemma B.76 $[\mathbf{c}_1]_{\varepsilon_1} \sqsubseteq \varepsilon_2[\mathbf{c}_2]$ if and only if $[\forall \mathbf{X}.\mathbf{c}_1]_{\forall \mathbf{X}.\varepsilon_1} \sqsubseteq \forall \mathbf{X}.\varepsilon_2[\forall \mathbf{X}.\mathbf{c}_2]$.

PROOF. We are required to prove that for all $(\forall \mathbf{X}.\mathbf{G}_1, \forall \mathbf{X}.\mathbf{c}_{11}, \forall \mathbf{X}.\mathbf{c}_{12}) \in \forall \mathbf{X}.\varepsilon_1$ there exists $(\forall \mathbf{X}.\mathbf{G}_2, \forall \mathbf{X}.\mathbf{c}_{21}, \forall \mathbf{X}.\mathbf{c}_{22}) \in \forall \mathbf{X}.\varepsilon_2$, \mathbf{c}_1 , \mathbf{c}_r and \mathbf{c}_{tt} such that

- $\mathbf{c}_{tt} : \forall \mathbf{X}.\mathbf{G}_1 \sqsubseteq \forall \mathbf{X}.\mathbf{G}_2$
- $\mathbf{c}_{tt}; \forall \mathbf{X}.\mathbf{c}_{21} = \mathbf{c}_1$
- $\forall \mathbf{X}.\mathbf{c}_{11}; \forall \mathbf{X}.\mathbf{c}_1 = \mathbf{c}_1$
- $\mathbf{c}_{tt}; \forall \mathbf{X}.\mathbf{c}_{22} = \mathbf{c}'_r$
- $\forall \mathbf{X}.\mathbf{c}_{12}; \forall \mathbf{X}.\mathbf{c}_2 = \mathbf{c}'_r$

We know by the definition of $\forall \mathbf{X}.\varepsilon_1$ that $(\mathbf{G}_1, \mathbf{c}_{11}, \mathbf{c}_{12}) \in \varepsilon_1$.

Also, we know by $[\mathbf{c}_1]_{\varepsilon_1} \sqsubseteq \varepsilon_2[\mathbf{c}_2]$ that for $(\mathbf{G}_1, \mathbf{c}_{11}, \mathbf{c}_{12}) \in \varepsilon_1$, there exist $(\mathbf{G}_2, \mathbf{c}_{21}, \mathbf{c}_{22}) \in \varepsilon_2$, \mathbf{c}_1 , \mathbf{c}'_2 , \mathbf{c}_t , such that

- $\mathbf{c}_t : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$
- $\mathbf{c}_t; \mathbf{c}_{21} = \mathbf{c}_1$
- $\mathbf{c}_{11}; \mathbf{c}_1 = \mathbf{c}_1$
- $\mathbf{c}_t; \mathbf{c}_{22} = \mathbf{c}'_2$
- $\mathbf{c}_{12}; \mathbf{c}_2 = \mathbf{c}'_2$

Therefore, we know that

- $\forall \mathbf{X}.\mathbf{c}_t : \forall \mathbf{X}.\mathbf{G}_1 \sqsubseteq \forall \mathbf{X}.\mathbf{G}_2$
- $\forall \mathbf{X}.\mathbf{c}_t; \forall \mathbf{X}.\mathbf{c}_{21} = \forall \mathbf{X}.\mathbf{c}_1$
- $\forall \mathbf{X}.\mathbf{c}_{11}; \forall \mathbf{X}.\mathbf{c}_1 = \forall \mathbf{X}.\mathbf{c}_1$
- $\forall \mathbf{X}.\mathbf{c}_t; \forall \mathbf{X}.\mathbf{c}_{22} = \forall \mathbf{X}.\mathbf{c}'_2$
- $\forall \mathbf{X}.\mathbf{c}_{12}; \forall \mathbf{X}.\mathbf{c}_2 = \forall \mathbf{X}.\mathbf{c}'_2$

Thus, taking $\mathbf{c}_1 = \forall \mathbf{X}.\mathbf{c}_1$, $\mathbf{c}_r = \forall \mathbf{X}.\mathbf{c}'_2$ and $\mathbf{c}_{tt} = \forall \mathbf{X}.\mathbf{c}_t$, the result follows immediately. \square

Lemma B.77 *If*

- $c' : G \sqsubseteq G'$
- $d : G' \sqsubseteq G''$
- $c';d = c$

then c is unique and $c : G \sqsubseteq G''$.

PROOF. We proceed by induction on the type precision transitivity rules. We have the following possible cases.

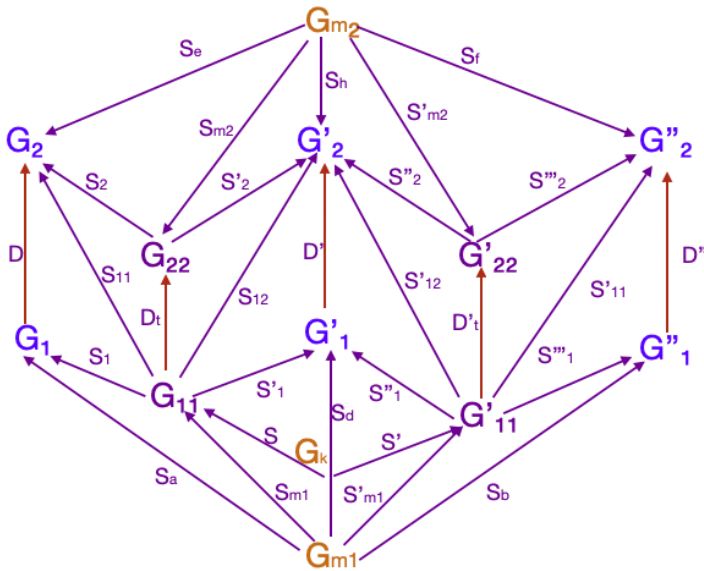
- If we have that $B;B = B$, where $c' = B$ and $d = B$, then the only possibility for c is $c = B$, by inspecting the rules of type precision transitivity, where $B : B \sqsubseteq B$. Thus, the result follows.
- If we have that $B;inj_B = inj_B$, where $c' = B$, $d = inj_B$ and $d : B \sqsubseteq ?_\delta$, then the only possibility for c is $c = inj_B$, by inspecting the rules of type precision transitivity, where $inj_B : B \sqsubseteq ?_\delta$. Thus, the result follows.
- If we have that $X;X = X$, where $c' = X$ and $d = X$, the only possibility for c is $c = X$, by inspecting the rules of type precision transitivity, where $X : X \sqsubseteq X$. Thus, the result follows.
- If we have that $F;inj_X = inj_X$, where $c' = F$, $d = inj_X$ and $d : F \sqsubseteq ?_\delta$, then the only possibility for c is $c = inj_X$, by inspecting the rules of type precision transitivity, where $inj_X : F \sqsubseteq ?_\delta$. Thus, the result follows.
- If $(c_1 \rightarrow c'_2);(d_1 \rightarrow d_2) = c_1 \rightarrow c_2$, where $c' = c_1 \rightarrow c'_2$, $d = d_1 \rightarrow d_2$, $c = c_1 \rightarrow c_2$, $G = G_1 \rightarrow G_2$, $G' = G'_1 \rightarrow G'_2$, $G'' = G''_1 \rightarrow G''_2$, $c_1 : G_1 \sqsubseteq G'_1$, $d_1 : G'_1 \sqsubseteq G''_1$, $c'_2 : G_2 \sqsubseteq G'_2$, $d_2 : G'_2 \sqsubseteq G''_2$, $c_1;d_1 = c_1$ and $c'_2;d_2 = c_2$, then, we know by the induction hypothesis on $c_1;d_1 = c_1$ and $c'_2;d_2 = c_2$ that c_1 and c_2 are unique and $c_1 : G_1 \sqsubseteq G''_1$ and $c_2 : G_2 \sqsubseteq G''_2$. Therefore, $c = c_1 \rightarrow c_2$ is unique and $c_1 \rightarrow c_2 : G_1 \rightarrow G_2 \sqsubseteq G''_1 \rightarrow G''_2$. Thus, the result follows.
- If $(\forall X.c_1);(\forall X.d_1) = \forall X.c_1$, where $c' = \forall X.c_1$, $d = \forall X.d_1$, $c = \forall X.c_1$, $G = \forall X.G_1$, $G' = \forall X.G'_1$, $G'' = \forall X.G''_1$, $c_1 : G_1 \sqsubseteq G'_1$, $d_1 : G'_1 \sqsubseteq G''_1$ and $c_1;d_1 = c_1$, then, we know by the induction hypothesis on $c_1;d_1 = c_1$ that c_1 is unique and $c_1 : G_1 \sqsubseteq G''_1$. Therefore, $c = \forall X.c_1$ is unique and $\forall X.c_1 : \forall X.G_1 \sqsubseteq \forall X.G''_1$. Thus, the result follows.
- If we have that $c';inj_? = c'$, where $d = inj_?$, $c' : G \sqsubseteq ?_\delta$ and $d : ?_\delta \sqsubseteq ?_\delta$, then the only possibility for c is $c = c'$, by inspecting the rules of type precision transitivity, where $c' : G \sqsubseteq ?_\delta$, by Lemma B.15. Thus, the result follows.
- If we have that $c';inj_{\rightarrow}(d') = inj_{\rightarrow}(c_1)$, where $d = inj_{\rightarrow}(d')$, $c';d' = c_1$, $c' : G \sqsubseteq G'$, $d : G' \sqsubseteq ?_\delta$ and $d' : G' \sqsubseteq ?_\delta \rightarrow ?_\delta$, then by the induction hypothesis on $c';d' = c_1$, we know that c_1 is unique and $c_1 : G \sqsubseteq ?_\delta \rightarrow ?_\delta$. Therefore, we know that $c = inj_{\rightarrow}(c_1)$ is unique and $c : G \sqsubseteq ?_\delta$. Thus, the result follows.
- If we have that $c';inj_{\forall}(d') = inj_{\forall}(c_1)$, where $d = inj_{\forall}(d')$, $c';d' = c_1$, $c' : G \sqsubseteq G'$, $d : G' \sqsubseteq ?_\delta$ and $d' : G' \sqsubseteq \forall X.?_\delta, X:X$, then by the induction hypothesis on $c';d' = c_1$, we know that c_1 is unique and $c_1 : G \sqsubseteq \forall X.?_\delta, X:X$. Therefore, we know that $c = inj_{\forall}(c_1)$ is unique and $c : G \sqsubseteq ?_\delta$. Thus, the result follows.

Lemma B.78 (Monotonicity of Consistent Transitivity) *If*

- $\varepsilon_1 : G_1 \sim G'_1$
- $\varepsilon'_1 : G'_1 \sim G''_1$
- $\varepsilon_2 : G_2 \sim G'_2$
- $\varepsilon'_2 : G'_2 \sim G''_2$
- $d : G_1 \sqsubseteq G_2$
- $d' : G'_1 \sqsubseteq G'_2$
- $d'' : G''_1 \sqsubseteq G''_2$
- $[d]\varepsilon_1 \sqsubseteq \varepsilon_2[d']$
- $[d']\varepsilon'_1 \sqsubseteq \varepsilon'_2[d'']$

then

- $[d](\varepsilon_1 \circledast \varepsilon'_1) \sqsubseteq (\varepsilon_2 \circledast \varepsilon'_2)[d'']$



PROOF. We are required to prove that $[d](\varepsilon_1 \circledast \varepsilon'_1) \sqsubseteq (\varepsilon_2 \circledast \varepsilon'_2)[d'']$, or what is the same that for all $(G_{m1}, c_a, c_b) \in \varepsilon_1 \circledast \varepsilon'_1$, there exists $(G_{m2}, c_e, d_f) \in \varepsilon_2 \circledast \varepsilon'_2, c_i, c'_i$ and d_t^* such that

- $d_t^* : G_{m1} \sqsubseteq G_{m2}$
- $d_t^*; c_e = c_i$

- $\mathbf{c}_a; \mathbf{d} = \mathbf{c}_i$
- $\mathbf{d}_t^*; \mathbf{c}_f = \mathbf{c}'_i$
- $\mathbf{c}_b; \mathbf{d}'' = \mathbf{c}'_i$

Let's suppose that $(\mathbf{G}_{m1}, \mathbf{c}_a, \mathbf{c}_b) \in (\mathbf{G}_{11}, \mathbf{c}_1, \mathbf{c}_1) \circ (\mathbf{G}'_{11}, \mathbf{c}''_1, \mathbf{c}'''_1)$, for some $(\mathbf{G}_{11}, \mathbf{c}_1, \mathbf{c}_1) \in \varepsilon_1$ and $(\mathbf{G}'_{11}, \mathbf{c}''_1, \mathbf{c}'''_1) \in \varepsilon'_1$.

A Then, we know that $pullback(\mathbf{G}'_1, (\mathbf{G}_{11}, \mathbf{c}_1), (\mathbf{G}'_{11}, \mathbf{c}''_1))$ must be defined. Therefore, there exists $(\mathbf{G}_{m1}, \mathbf{c}_{m1}, \mathbf{c}'_{m1}, \mathbf{c}_d) = pullback(\mathbf{G}'_1, (\mathbf{G}_{11}, \mathbf{c}_1), (\mathbf{G}'_{11}, \mathbf{c}''_1))$ such that (Definition B.16):

- A.1) $\mathbf{c}_{m1} : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}_{11}$
- A.2) $\mathbf{c}'_{m1} : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}'_{11}$
- A.3) $\mathbf{c}_d : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}'_1$
- A.4) $\mathbf{c}_{m1}; \mathbf{c}'_1 = \mathbf{c}_d$
- A.5) $\mathbf{c}'_{m1}; \mathbf{c}''_1 = \mathbf{c}_d$

B We also know that (by definition of consistent transitivity):

- B.1) $\mathbf{c}_a : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}_1$
- B.2) $\mathbf{c}_b : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}''_1$
- B.3) $\mathbf{c}_{m1}; \mathbf{c}_1 = \mathbf{c}_a$
- B.4) $\mathbf{c}'_{m1}; \mathbf{c}'''_1 = \mathbf{c}_b$

C Since $[\mathbf{d}]_{\varepsilon_1} \sqsubseteq \varepsilon_2[\mathbf{d}']$ and $[\mathbf{d}']_{\varepsilon'_1} \sqsubseteq \varepsilon'_2[\mathbf{d}'']$, by Definition 6.11, we know that for $(\mathbf{G}_{11}, \mathbf{c}_1, \mathbf{c}_1) \in \varepsilon_1$ and $(\mathbf{G}'_{11}, \mathbf{c}''_1, \mathbf{c}'''_1) \in \varepsilon'_1$, there exists $(\mathbf{G}_{22}, \mathbf{c}_2, \mathbf{c}'_2) \in \varepsilon_2$, \mathbf{c}_{11} , \mathbf{c}_{12} and \mathbf{d}_t , and $(\mathbf{G}'_{22}, \mathbf{c}''_2, \mathbf{c}'''_2) \in \varepsilon'_2$, \mathbf{c}'_{11} , \mathbf{c}'_{12} and \mathbf{d}'_t such that:

- C.1) $\mathbf{d}_t : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{22}$
- C.2) $\mathbf{c}_{11} : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_2$
- C.3) $\mathbf{c}_{12} : \mathbf{G}_{11} \sqsubseteq \mathbf{G}'_2$
- C.4) $\mathbf{d}_t; \mathbf{c}_2 = \mathbf{c}_{11}$
- C.5) $\mathbf{c}_1; \mathbf{d} = \mathbf{c}_{11}$
- C.6) $\mathbf{d}_t; \mathbf{c}'_2 = \mathbf{c}_{12}$
- C.7) $\mathbf{c}'_1; \mathbf{d}' = \mathbf{c}_{12}$
- C.8) $\mathbf{d}'_t : \mathbf{G}'_{11} \sqsubseteq \mathbf{G}'_{22}$
- C.9) $\mathbf{c}'_{12} : \mathbf{G}'_{11} \sqsubseteq \mathbf{G}'_2$
- C.10) $\mathbf{c}'_{11} : \mathbf{G}'_{11} \sqsubseteq \mathbf{G}''_2$
- C.11) $\mathbf{d}'_t; \mathbf{c}''_2 = \mathbf{c}'_{12}$
- C.12) $\mathbf{c}''_1; \mathbf{d}' = \mathbf{c}'_{12}$
- C.13) $\mathbf{d}'_t; \mathbf{c}'''_2 = \mathbf{c}'_{11}$
- C.14) $\mathbf{c}'''_1; \mathbf{d}'' = \mathbf{c}'_{11}$

D By (A.3), $\mathbf{d}' : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$, Lemma B.20 and Lemma B.77, we get that there exists a unique \mathbf{c}_1 such that

$$\text{D.1) } \mathbf{c}_1 : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}'_2$$

$$\text{D.2) } \mathbf{c}_d; \mathbf{d}' = \mathbf{c}_1$$

E By (A.4), (C.7) and Lemma B.13, we get that

$$\text{E.1) } \mathbf{c}_{m1}; \mathbf{c}_{12} = \mathbf{c}_1$$

F By (C.6) and (E.1), we get that $\mathbf{c}_{m1}; (\mathbf{d}_t; \mathbf{c}'_2) = \mathbf{c}_1$. By Lemma B.13, we get that $(\mathbf{c}_{m1}; \mathbf{d}_t); \mathbf{c}'_2 = \mathbf{c}_1$. By Definition B.20, we know that exists \mathbf{c}_4 such that

$$\text{F.1) } \mathbf{c}_{m1}; \mathbf{d}_t = \mathbf{c}_4$$

$$\text{F.2) } \mathbf{c}_4; \mathbf{c}'_2 = \mathbf{c}_1$$

$$\text{F.3) } \mathbf{c}_4 : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}_{22}$$

G By (A.5), (C.12) and Lemma B.13, we get that

$$\text{G.1) } \mathbf{c}'_{m1}; \mathbf{c}'_{12} = \mathbf{c}_1$$

H By (C.11) and (G.1), we get that $\mathbf{c}'_{m1}; (\mathbf{d}'_t; \mathbf{c}''_2) = \mathbf{c}_1$. By Lemma B.13, we get that $(\mathbf{c}'_{m1}; \mathbf{d}'_t); \mathbf{c}''_2 = \mathbf{c}_1$. By Definition B.20, we know that exists \mathbf{c}'_4 such that

$$\text{H.1) } \mathbf{c}'_{m1}; \mathbf{d}'_t = \mathbf{c}'_4$$

$$\text{H.2) } \mathbf{c}'_4; \mathbf{c}''_2 = \mathbf{c}_1$$

$$\text{H.3) } \mathbf{c}'_4 : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}'_{22}$$

I By $\mathbf{c}'_2 : \mathbf{G}_{22} \sqsubseteq \mathbf{G}'_2$, $\mathbf{c}''_2 : \mathbf{G}'_{22} \sqsubseteq \mathbf{G}'_2$, (F.3), (H.3), (F.2) and (H.2), we make use of the universal property (Lemma 6.3), this means that there exist $(\mathbf{G}_{m2}, \mathbf{c}_{m2}, \mathbf{c}'_{m2}, \mathbf{c}_h) = \text{pullback}(\mathbf{G}'_2, (\mathbf{G}_{22}, \mathbf{c}'_2), (\mathbf{G}'_{22}, \mathbf{c}''_2))$ and \mathbf{c}_k such that

$$\text{I.1) } \mathbf{c}_k : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}_{m2}$$

$$\text{I.2) } \mathbf{c}_k; \mathbf{c}_{m2} = \mathbf{c}_4$$

$$\text{I.3) } \mathbf{c}_k; \mathbf{c}'_{m2} = \mathbf{c}'_4$$

$$\text{I.4) } \mathbf{c}_k; \mathbf{c}_h = \mathbf{c}_1$$

$$\text{I.5) } \mathbf{c}_{m2}; \mathbf{c}'_2 = \mathbf{c}_h$$

$$\text{I.6) } \mathbf{c}'_{m2}; \mathbf{c}''_2 = \mathbf{c}_h$$

$$\text{I.7) } \mathbf{c}_h : \mathbf{G}_{m2} \sqsubseteq \mathbf{G}'_2$$

J By (B.1), $\mathbf{d} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, Lemma B.20 and Lemma B.77, there exists a unique \mathbf{c}_i such that

$$\text{J.1) } \mathbf{c}_i : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}_2$$

$$\text{J.2) } \mathbf{c}_a; \mathbf{d} = \mathbf{c}_i$$

K By (B.3) and (J.2), we get that $(\mathbf{c}_{m1}; \mathbf{c}_1); \mathbf{d} = \mathbf{c}_i$. By Lemma B.13, we know that $\mathbf{c}_{m1}; (\mathbf{c}_1; \mathbf{d}) = \mathbf{c}_i$. By (C.5) and Lemma B.77, we know that \mathbf{c}_{11} is unique and therefore

$$\text{K.1) } \mathbf{c}_{m1}; \mathbf{c}_{11} = \mathbf{c}_i$$

L By (C.4) and (K.1), we know that $\mathbf{c}_{m1};(\mathbf{d}_t;\mathbf{c}_2) = \mathbf{c}_i$. By Lemma B.13, we get that $(\mathbf{c}_{m1};\mathbf{d}_t);\mathbf{c}_2 = \mathbf{c}_i$. By (F.1) and Lemma B.77, we know that \mathbf{c}_4 is unique and therefore

$$\text{L.1) } \mathbf{c}_4;\mathbf{c}_2 = \mathbf{c}_i$$

M By (I.2) and (L.1), we know that $(\mathbf{c}_k;\mathbf{c}_{m2});\mathbf{c}_2 = \mathbf{c}_i$. By Lemma B.13, we know that $\mathbf{c}_k;(\mathbf{c}_{m2};\mathbf{c}_2) = \mathbf{c}_i$. By Definition B.21, we get that there exists \mathbf{c}_e such that

$$\text{M.1) } \mathbf{c}_e : \mathbf{G}_{m2} \sqsubseteq \mathbf{G}_2$$

$$\text{M.2) } \mathbf{c}_{m2};\mathbf{c}_2 = \mathbf{c}_e$$

$$\text{M.3) } \mathbf{c}_k;\mathbf{c}_e = \mathbf{c}_i$$

N By (B.2), $\mathbf{d}'' : \mathbf{G}_1'' \sqsubseteq \mathbf{G}_2''$, Lemma B.20 and Lemma B.77, there exists a unique \mathbf{c}'_i such that

$$\text{N.1) } \mathbf{c}'_i : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}_2''$$

$$\text{N.2) } \mathbf{c}_b;\mathbf{d}'' = \mathbf{c}'_i$$

O By (B.4) and (N.2), we get that $(\mathbf{c}'_{m1};\mathbf{c}'_1''');\mathbf{d}'' = \mathbf{c}'_i$. By Lemma B.13, we know that $\mathbf{c}'_{m1};(\mathbf{c}'_1''';\mathbf{d}'') = \mathbf{c}'_i$. By (C.14) and Lemma B.77, we know that \mathbf{c}'_{11} is unique and therefore

$$\text{O.1) } \mathbf{c}'_{m1};\mathbf{c}'_{11} = \mathbf{c}'_i$$

P By (C.13) and (O.1), we know that $\mathbf{c}'_{m1};(\mathbf{d}'_t;\mathbf{c}'_2''') = \mathbf{c}'_i$. By Lemma B.13, we get that $(\mathbf{c}'_{m1};\mathbf{d}'_t);\mathbf{c}'_2'' = \mathbf{c}'_i$. By (H.1) and Lemma B.77, we know that \mathbf{c}'_4 is unique and therefore

$$\text{P.1) } \mathbf{c}'_4;\mathbf{c}'_2'' = \mathbf{c}'_i$$

Q By (I.3) and (P.1), we know that $(\mathbf{c}_k;\mathbf{c}'_{m2});\mathbf{c}'_2'' = \mathbf{c}'_i$. By Lemma B.13, we know that $\mathbf{c}_k;(\mathbf{c}'_{m2};\mathbf{c}'_2'') = \mathbf{c}'_i$. By Definition B.21, we get that there exists \mathbf{c}_f such that

$$\text{Q.1) } \mathbf{c}_f : \mathbf{G}_{m2} \sqsubseteq \mathbf{G}_2''$$

$$\text{Q.2) } \mathbf{c}'_{m2};\mathbf{c}'_2'' = \mathbf{c}_f$$

$$\text{Q.3) } \mathbf{c}_k;\mathbf{c}_f = \mathbf{c}'_i$$

Note that $(\mathbf{G}_{m2}, \mathbf{c}_e, \mathbf{c}_f) \in \varepsilon_2 \ ; \ \varepsilon'_2$. More specifically $(\mathbf{G}_{m2}, \mathbf{c}_e, \mathbf{c}_f) \in (\mathbf{G}_{22}, \mathbf{c}_2, \mathbf{c}'_2) \ ; \ (\mathbf{G}'_{22}, \mathbf{c}_2'', \mathbf{c}'_2''')$, where $(\mathbf{G}_{22}, \mathbf{c}_2, \mathbf{c}'_2) \in \varepsilon_2$ and $(\mathbf{G}'_{22}, \mathbf{c}_2'', \mathbf{c}'_2''') \in \varepsilon'_2$.

R Since (B.1) $(\mathbf{c}_a : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}_1)$, $\mathbf{d} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, (I.1) $(\mathbf{c}_k : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}_{m2})$, (M.1) $(\mathbf{c}_e : \mathbf{G}_{m2} \sqsubseteq \mathbf{G}_2)$, (Q.1) $(\mathbf{c}_f : \mathbf{G}_{m2} \sqsubseteq \mathbf{G}_2'')$, (B.2) $(\mathbf{c}_b : \mathbf{G}_{m1} \sqsubseteq \mathbf{G}_1'')$, $\mathbf{d}'' : \mathbf{G}_1'' \sqsubseteq \mathbf{G}_2''$, taking $\mathbf{d}_t^* = \mathbf{c}_k$, we get that

- $\mathbf{c}_k;\mathbf{c}_e = \mathbf{c}_i$ (M.3)
- $\mathbf{c}_a;\mathbf{d} = \mathbf{c}_i$ (J.2)
- $\mathbf{c}_k;\mathbf{c}_f = \mathbf{c}'_i$ (Q.3)
- $\mathbf{c}_b;\mathbf{d}'' = \mathbf{c}'_i$ (N.2)

Thus, the result holds. □

Theorem B.79 Suppose $\vdash \mathbf{t}_1 : \mathbf{G}_1$ and $\vdash \mathbf{c} : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$. Then,

- $\vdash t_2 : G_2$ and $c : G_1 \sqsubseteq G_2$.
- $t_1 \xrightarrow{*} v_1$ implies $t_2 \xrightarrow{*} v_2$ and $\vdash c : v_1 \sqsubseteq v_2$.
- t_1 diverges implies t_2 diverges.

PROOF. The proof is by case analysis on the reduction of t_1 or t_2 .

- Suppose that $t_1 \xrightarrow{*} v_1$. Then, we get that $t_2 \xrightarrow{*} v_2$, $\vdash c : v_1 \sqsubseteq v_2$, by Lemma B.88, and the result holds.
- Suppose that t_1 diverges. Then, we get that t_2 diverges, by Lemma B.88, and the result holds.

□

Lemma B.80 *If $\vdash c : v_1 \sqsubseteq t_2$ then $t_2 = v_2$.*

Lemma B.81 *If $c : G_1 \sqsubseteq G_2$ then $c[X \mapsto F] : G_1[F/X] \sqsubseteq G_2[F/X]$.*

PROOF. Follow by induction on $c : G_1 \sqsubseteq G_2$.

Case ($B : B \sqsubseteq B$). The results follows immediately due to $B[X \mapsto F] = B$, $B[F/X] = B$ and $B : B \sqsubseteq B$.

Case ($Y : Y \sqsubseteq Y$). We have the following cases.

- $Y \not\rightarrow X$ The results follows immediately due to $Y[X \mapsto F] = Y$, $Y[F/X] = Y$ and $Y : Y \sqsubseteq Y$.
- $Y = X$. The results follows immediately due to $X[X \mapsto F] = F$, $X[F/X] = F$ and $F : F \sqsubseteq F$.

Case ($c_1 \rightarrow c_2 : G_{11} \rightarrow G_{12} \sqsubseteq G_{21} \rightarrow G_{22}$). We know that

$$\frac{c_1 : G_{11} \sqsubseteq G_{21} \quad c_2 : G_{12} \sqsubseteq G_{22}}{c_1 \rightarrow c_2 : G_{11} \rightarrow G_{12} \sqsubseteq G_{21} \rightarrow G_{22}}$$

Then, we know that $c_1 : G_{11} \sqsubseteq G_{21}$ and $c_2 : G_{12} \sqsubseteq G_{22}$. We know that

- $(G_{11} \rightarrow G_{12})[F/X] = (G_{11}[X/F] \rightarrow G_{12}[X/F])$
- $(G_{21} \rightarrow G_{22})[F/X] = (G_{21}[X/F] \rightarrow G_{22}[X/F])$

Thus, we are required to prove that

- $c_1[X \mapsto F] \rightarrow c_2[X \mapsto F] : (G_{11}[X/F] \rightarrow G_{12}[X/F]) \sqsubseteq (G_{21}[X/F] \rightarrow G_{22}[X/F])$

Or what is the same, we are required to prove that

- $c_1 : G_{11}[X/F] \sqsubseteq G_{21}[X/F]$

- $c_2 : G_{12}[X/F] \sqsubseteq G_{22}[X/F]$

But the result follows immediately by the induction hypothesis on $c_1 : G_{11} \sqsubseteq G_{21}$ and $c_2 : G_{12} \sqsubseteq G_{22}$.

Case ($c_1 \times c_2 : G_{11} \times G_{12} \sqsubseteq G_{21} \times G_{22}$). We know that

$$\frac{c_1 : G_{11} \sqsubseteq G_{21} \quad c_2 : G_{12} \sqsubseteq G_{22}}{c_1 \times c_2 : G_{11} \times G_{12} \sqsubseteq G_{21} \times G_{22}}$$

Then, we know that $c_1 : G_{11} \sqsubseteq G_{21}$ and $c_2 : G_{12} \sqsubseteq G_{22}$. We know that

- $(G_{11} \times G_{12})[F/X] = (G_{11}[X/F] \times G_{12}[X/F])$
- $(G_{21} \times G_{22})[F/X] = (G_{21}[X/F] \times G_{22}[X/F])$

Thus, we are required to prove that

- $c_1[X \mapsto F] \times c_2[X \mapsto F] : (G_{11}[X/F] \times G_{12}[X/F]) \sqsubseteq (G_{21}[X/F] \times G_{22}[X/F])$

Or what is the same, we are required to prove that

- $c_1 : G_{11}[X/F] \sqsubseteq G_{21}[X/F]$
- $c_2 : G_{12}[X/F] \sqsubseteq G_{22}[X/F]$

But the result follows immediately by the induction hypothesis on $c_1 : G_{11} \sqsubseteq G_{21}$ and $c_2 : G_{12} \sqsubseteq G_{22}$.

Case ($\forall Y.c_1 : \forall Y.G_{11} \sqsubseteq \forall Y.G_{21}$). We know that

$$\frac{c_1 : G_{11} \sqsubseteq G_{21}}{\forall Y.c_1 : \forall Y.G_{11} \sqsubseteq \forall Y.G_{21}}$$

Then, we know that $c_1 : G_{11} \sqsubseteq G_{21}$. We know that

- $(\forall Y.G_{11})[F/X] = (\forall Y.G_{11}[X/F])$
- $(\forall Y.G_{21})[F/X] = (\forall Y.G_{21}[X/F])$

Thus, we are required to prove that

- $\forall Y.c_1[X \mapsto F] : (\forall Y.G_{11}[X/F]) \sqsubseteq (\forall Y.G_{21}[X/F])$

Or what is the same, we are required to prove that

- $c_1 : G_{11}[X/F] \sqsubseteq G_{21}[X/F]$

But the result follows immediately by the induction hypothesis on $c_1 : G_{11} \sqsubseteq G_{21}$.

Case ($\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta}$). The results follows immediately due to $\mathbf{B}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}$, $\mathbf{B}[\mathbf{F}/\mathbf{X}] = \mathbf{B}$ and $\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta}[\mathbf{F}/\mathbf{X}]$.

Case ($\text{inj}_{\mathbf{Y}} : \mathbf{F}' \sqsubseteq ?_{\delta}$). We know that $\mathbf{Y} : \mathbf{F}' \in \delta$ and therefore, $\mathbf{Y} : \mathbf{F}'[\mathbf{F}/\mathbf{X}] \in \delta[\mathbf{F}/\mathbf{X}]$. Thus, we get that $\text{inj}_{\mathbf{Y}} : \mathbf{F}'[\mathbf{F}/\mathbf{X}] \sqsubseteq ?_{\delta}[\mathbf{F}/\mathbf{X}]$, as we are required to show.

Case ($\text{inj}_{\rightarrow}(\mathbf{c}') : \mathbf{G}_1 \sqsubseteq ?_{\delta}$). We know that $\mathbf{c}' : \mathbf{G}_1 \sqsubseteq ?_{\delta} \rightarrow ?_{\delta}$. By the inductive hypothesis, we get that $\mathbf{c}'[\mathbf{F}/\mathbf{X}] : \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \sqsubseteq ?_{\delta}[\mathbf{F}/\mathbf{X}] \rightarrow ?_{\delta}[\mathbf{F}/\mathbf{X}]$. Therefore, we get that $\text{inj}_{\rightarrow}(\mathbf{c}'[\mathbf{F}/\mathbf{X}]) : \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \sqsubseteq ?_{\delta}[\mathbf{F}/\mathbf{X}]$, and the result holds.

Case ($\text{inj}_{\times}(\mathbf{c}') : \mathbf{G}_1 \sqsubseteq ?_{\delta}$). We know that $\mathbf{c}' : \mathbf{G}_1 \sqsubseteq ?_{\delta} \times ?_{\delta}$. By the inductive hypothesis, we get that $\mathbf{c}'[\mathbf{F}/\mathbf{X}] : \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \sqsubseteq ?_{\delta}[\mathbf{F}/\mathbf{X}] \times ?_{\delta}[\mathbf{F}/\mathbf{X}]$. Therefore, we get that $\text{inj}_{\times}(\mathbf{c}'[\mathbf{F}/\mathbf{X}]) : \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \sqsubseteq ?_{\delta}[\mathbf{F}/\mathbf{X}]$, and the result holds.

Case ($\text{inj}_{\forall}(\mathbf{c}') : \mathbf{G}_1 \sqsubseteq ?_{\delta}$). We know that $\mathbf{c}' : \mathbf{G}_1 \sqsubseteq \forall \mathbf{Y}. ?_{\delta, \mathbf{Y}: \mathbf{Y}}$. By the inductive hypothesis, we get that $\mathbf{c}'[\mathbf{F}/\mathbf{X}] : \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \sqsubseteq \forall \mathbf{Y}. ?_{\delta, \mathbf{Y}: \mathbf{Y}}[\mathbf{F}/\mathbf{X}]$. Therefore, we get that $\text{inj}_{\forall}(\mathbf{c}'[\mathbf{F}/\mathbf{X}]) : \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \sqsubseteq ?_{\delta}[\mathbf{F}/\mathbf{X}]$, and the result holds.

Case ($\text{inj}_{?} : ?_{\delta} \sqsubseteq ?_{\delta'}$). The results follows immediately due to $\delta[\mathbf{F}/\mathbf{X}] \subseteq \delta'[\mathbf{F}/\mathbf{X}]$ and $\text{inj}_{?} : ?_{\delta}[\mathbf{F}/\mathbf{X}] \sqsubseteq ?_{\delta'}[\mathbf{F}/\mathbf{X}]$.

□

Lemma B.82 *If $[\mathbf{c}]_{\varepsilon_1} \sqsubseteq \varepsilon_2[\mathbf{c}']$ then $[\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]]_{\varepsilon_1}[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \varepsilon_2[\mathbf{X} \mapsto \mathbf{F}][\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}]]$.*

PROOF. Since $[\mathbf{c}]_{\varepsilon_1} \sqsubseteq \varepsilon_2[\mathbf{c}']$, we know that for all $(\mathbf{G}_1'', \mathbf{c}_1, \mathbf{c}_1) \in \varepsilon_1$, there exists $(\mathbf{G}_2'', \mathbf{c}_2, \mathbf{c}_2') \in \varepsilon_2$, \mathbf{c}_1'' , \mathbf{c}_2'' and \mathbf{c}_t such that

- $\mathbf{c}_t : \mathbf{G}_1'' \sqsubseteq \mathbf{G}_2''$
- $\mathbf{c}_t; \mathbf{c}_2 = \mathbf{c}_1''$
- $\mathbf{c}_1; \mathbf{c} = \mathbf{c}_1''$
- $\mathbf{c}_t; \mathbf{c}_2' = \mathbf{c}_2''$
- $\mathbf{c}_1'; \mathbf{c}' = \mathbf{c}_2''$

We are required to prove that $[\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}]]_{\varepsilon_1}[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \varepsilon_2[\mathbf{X} \mapsto \mathbf{F}][\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}]]$. We proceed by proving that for all $\mathbf{S}_1 = (\mathbf{G}_1''[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]) \in \varepsilon_1[\mathbf{X} \mapsto \mathbf{F}]$, there exists $\mathbf{S}_2 = (\mathbf{G}_2''[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}], \mathbf{c}_2'[\mathbf{X} \mapsto \mathbf{F}]) \in \varepsilon_2[\mathbf{X} \mapsto \mathbf{F}]$, $\mathbf{c}_1''[\mathbf{X} \mapsto \mathbf{F}]$, $\mathbf{c}_2''[\mathbf{X} \mapsto \mathbf{F}]$ and $\mathbf{c}_t[\mathbf{X} \mapsto \mathbf{F}]$ such that

- $\mathbf{c}_t[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}_1''[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}_2''[\mathbf{X} \mapsto \mathbf{F}]$
- $\mathbf{c}_t[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_1''[\mathbf{X} \mapsto \mathbf{F}]$
- $\mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_1''[\mathbf{X} \mapsto \mathbf{F}]$
- $\mathbf{c}_t[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}_2'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_2''[\mathbf{X} \mapsto \mathbf{F}]$
- $\mathbf{c}_1'[\mathbf{X} \mapsto \mathbf{F}]; \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{c}_2''[\mathbf{X} \mapsto \mathbf{F}]$

The result holds immediately by Lemmas B.81 and B.66.

□

Lemma B.83 *If $[\mathbf{c}_1 \rightarrow \mathbf{c}'_1] \varepsilon_1 \sqsubseteq \varepsilon_2 [\mathbf{c}_2 \rightarrow \mathbf{c}'_2]$ then*

- $[\mathbf{c}_2] \text{dom}(\varepsilon_1) \sqsubseteq \text{dom}(\varepsilon_2) [\mathbf{c}_1]$
- $[\mathbf{c}'_1] \text{cod}(\varepsilon_1) \sqsubseteq \text{cod}(\varepsilon_2) [\mathbf{c}'_2]$

PROOF. Since $[\mathbf{c}_1 \rightarrow \mathbf{c}_1] \varepsilon_1 \sqsubseteq \varepsilon_2 [\mathbf{c}_2 \rightarrow \mathbf{c}'_2]$, we know that for all $\mathbf{S}_1 \in \varepsilon_1$, $\mathbf{S}_1 = (\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \mathbf{c}'_{11} \rightarrow \mathbf{c}'_{12}) \in \varepsilon_1$ and there exists $\mathbf{S}_2 = (\mathbf{G}_{21} \rightarrow \mathbf{G}_{22}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22}, \mathbf{c}'_{21} \rightarrow \mathbf{c}'_{22}) \in \varepsilon_2$, $\mathbf{c}''_{11} \rightarrow \mathbf{c}''_{12}$, $\mathbf{c}''_{21} \rightarrow \mathbf{c}''_{22}$ and $\mathbf{c}_t \rightarrow \mathbf{c}'_t$ such that

- $\mathbf{c}_t \rightarrow \mathbf{c}'_t : \mathbf{G}_{11} \rightarrow \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}$
- $\mathbf{c}_t \rightarrow \mathbf{c}'_t ; \mathbf{c}_{21} \rightarrow \mathbf{c}_{22} = \mathbf{c}''_{11} \rightarrow \mathbf{c}''_{12}$
- $\mathbf{c}_{11} \rightarrow \mathbf{c}_{12} ; \mathbf{c}_1 \rightarrow \mathbf{c}_1 = \mathbf{c}''_{11} \rightarrow \mathbf{c}''_{12}$
- $\mathbf{c}_t \rightarrow \mathbf{c}'_t ; \mathbf{c}'_{21} \rightarrow \mathbf{c}'_{22} = \mathbf{c}''_{21} \rightarrow \mathbf{c}''_{22}$
- $\mathbf{c}'_{11} \rightarrow \mathbf{c}'_{12} ; \mathbf{c}_2 \rightarrow \mathbf{c}'_2 = \mathbf{c}''_{21} \rightarrow \mathbf{c}''_{22}$

We know that

- $\mathbf{c}_t : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$
- $\mathbf{c}_t ; \mathbf{c}_{21} = \mathbf{c}''_{11}$
- $\mathbf{c}_{11} ; \mathbf{c}_1 = \mathbf{c}''_{11}$
- $\mathbf{c}_t ; \mathbf{c}'_{21} = \mathbf{c}''_{21}$
- $\mathbf{c}'_{11} ; \mathbf{c}_2 = \mathbf{c}''_{21}$
- $\mathbf{c}'_t : \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$
- $\mathbf{c}'_t ; \mathbf{c}_{22} = \mathbf{c}''_{12}$
- $\mathbf{c}_{12} ; \mathbf{c}_1 = \mathbf{c}''_{12}$
- $\mathbf{c}'_t ; \mathbf{c}'_{22} = \mathbf{c}''_{22}$
- $\mathbf{c}'_{12} ; \mathbf{c}'_2 = \mathbf{c}''_{22}$

We know that $\text{dom}(\{(\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \mathbf{c}'_{11} \rightarrow \mathbf{c}'_{12})\}) = \{(\mathbf{G}_{11}, \mathbf{c}'_{11}, \mathbf{c}_{11})\}$, $\text{dom}(\{(\mathbf{G}_{21} \rightarrow \mathbf{G}_{22}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22}, \mathbf{c}'_{21} \rightarrow \mathbf{c}'_{22})\}) = \{(\mathbf{G}_{21}, \mathbf{c}'_{21}, \mathbf{c}_{21})\}$, $\text{cod}(\{(\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}, \mathbf{c}_{11} \rightarrow \mathbf{c}_{12}, \mathbf{c}'_{11} \rightarrow \mathbf{c}'_{12})\}) = \{(\mathbf{G}_{12}, \mathbf{c}_{12}, \mathbf{c}'_{12})\}$ and $\text{cod}(\{(\mathbf{G}_{21} \rightarrow \mathbf{G}_{22}, \mathbf{c}_{21} \rightarrow \mathbf{c}_{22}, \mathbf{c}'_{21} \rightarrow \mathbf{c}'_{22})\}) = \{(\mathbf{G}_{22}, \mathbf{c}_{22}, \mathbf{c}'_{22})\}$. Thus, the result holds immediately. \square

Lemma B.84 *If $[\mathbf{c}_1 \times \mathbf{c}'_1] \varepsilon_1 \sqsubseteq \varepsilon_2 [\mathbf{c}_2 \times \mathbf{c}'_2]$ then*

- $[\mathbf{c}_1] \pi_1(\varepsilon_1) \sqsubseteq \pi_1(\varepsilon_2) [\mathbf{c}_2]$
- $[\mathbf{c}'_1] \pi_2(\varepsilon_1) \sqsubseteq \pi_2(\varepsilon_2) [\mathbf{c}'_2]$

PROOF. Since $[\mathbf{c}_1 \times \mathbf{c}_1]_{\varepsilon_1} \sqsubseteq \varepsilon_2[\mathbf{c}_2 \times \mathbf{c}'_2]$, we know that for all $\mathbf{S}_1 \in \varepsilon_1$, $\mathbf{S}_1 = (\mathbf{G}_{11} \times \mathbf{G}_{12}, \mathbf{c}_{11} \times \mathbf{c}_{12}, \mathbf{c}'_{11} \times \mathbf{c}'_{12}) \in \varepsilon_1$ and there exists $\mathbf{S}_2 = (\mathbf{G}_{21} \times \mathbf{G}_{22}, \mathbf{c}_{21} \times \mathbf{c}_{22}, \mathbf{c}'_{21} \times \mathbf{c}'_{22}) \in \varepsilon_2$, $\mathbf{c}''_{11} \times \mathbf{c}''_{12}$, $\mathbf{c}''_{21} \times \mathbf{c}''_{22}$ and $\mathbf{c}_t \times \mathbf{c}'_t$ such that

- $\mathbf{c}_t \times \mathbf{c}'_t : \mathbf{G}_{11} \times \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{21} \times \mathbf{G}_{22}$
- $\mathbf{c}_t \times \mathbf{c}'_t; \mathbf{c}_{21} \times \mathbf{c}_{22} = \mathbf{c}''_{11} \times \mathbf{c}''_{12}$
- $\mathbf{c}_{11} \times \mathbf{c}_{12}; \mathbf{c}_1 \times \mathbf{c}_1 = \mathbf{c}''_{11} \times \mathbf{c}''_{12}$
- $\mathbf{c}_t \times \mathbf{c}'_t; \mathbf{c}'_{21} \times \mathbf{c}'_{22} = \mathbf{c}''_{21} \times \mathbf{c}''_{22}$
- $\mathbf{c}'_{11} \times \mathbf{c}'_{12}; \mathbf{c}_2 \times \mathbf{c}'_2 = \mathbf{c}''_{21} \times \mathbf{c}''_{22}$

We know that

- $\mathbf{c}_t : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$
- $\mathbf{c}_t; \mathbf{c}_{21} = \mathbf{c}''_{11}$
- $\mathbf{c}_{11}; \mathbf{c}_1 = \mathbf{c}''_{11}$
- $\mathbf{c}_t; \mathbf{c}'_{21} = \mathbf{c}''_{21}$
- $\mathbf{c}'_{11}; \mathbf{c}_2 = \mathbf{c}''_{21}$
- $\mathbf{c}'_t : \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$
- $\mathbf{c}'_t; \mathbf{c}_{22} = \mathbf{c}''_{12}$
- $\mathbf{c}_{12}; \mathbf{c}_1 = \mathbf{c}''_{12}$
- $\mathbf{c}'_t; \mathbf{c}'_{22} = \mathbf{c}''_{22}$
- $\mathbf{c}'_{12}; \mathbf{c}'_2 = \mathbf{c}''_{22}$

We know that $\pi_1(\{(\mathbf{G}_{11} \times \mathbf{G}_{12}, \mathbf{c}_{11} \times \mathbf{c}_{12}, \mathbf{c}'_{11} \times \mathbf{c}'_{12})\}) = \{(\mathbf{G}_{11}, \mathbf{c}'_{11}, \mathbf{c}_{11})\}$, $\pi_1(\{(\mathbf{G}_{21} \times \mathbf{G}_{22}, \mathbf{c}_{21} \times \mathbf{c}_{22}, \mathbf{c}'_{21} \times \mathbf{c}'_{22})\}) = \{(\mathbf{G}_{21}, \mathbf{c}'_{21}, \mathbf{c}_{21})\}$, $\pi_2(\{(\mathbf{G}_{11} \times \mathbf{G}_{12}, \mathbf{c}_{11} \times \mathbf{c}_{12}, \mathbf{c}'_{11} \times \mathbf{c}'_{12})\}) = \{(\mathbf{G}_{12}, \mathbf{c}_{12}, \mathbf{c}'_{12})\}$ and $\pi_2(\{(\mathbf{G}_{21} \times \mathbf{G}_{22}, \mathbf{c}_{21} \times \mathbf{c}_{22}, \mathbf{c}'_{21} \times \mathbf{c}'_{22})\}) = \{(\mathbf{G}_{22}, \mathbf{c}_{22}, \mathbf{c}'_{22})\}$. Thus, the result holds immediately. \square

Lemma B.85 (Monotonicity of Type Substitution) *If $[\forall \mathbf{X}. \mathbf{c}']_{\varepsilon_1} \sqsubseteq \varepsilon_2[\forall \mathbf{X}. \mathbf{c}]$, then $[\mathbf{c}'](schm(\varepsilon_1)) \sqsubseteq (schm(\varepsilon_2))[\mathbf{c}]$ and $[\mathbf{c}'[\mathbf{F}/\mathbf{X}]](schm(\varepsilon_1))[\mathbf{F}/\mathbf{X}] \sqsubseteq (schm(\varepsilon_2))[\mathbf{F}/\mathbf{X}][\mathbf{c}[\mathbf{F}/\mathbf{X}]]$.*

PROOF. Similar to the function case. \square

Lemma B.86 *If $\vdash \mathbf{c} : \mathbf{v}_1 \sqsubseteq \mathbf{t}_2$ then $\mathbf{t}_2 = \mathbf{v}_2$.*

PROOF. Exploring $\vdash \mathbf{c} : \mathbf{v}_1 \sqsubseteq \mathbf{t}_2$ rules. \square

Lemma B.87 *If $\mathbf{c} : \mathbf{G} \sqsubseteq \mathbf{G}'$ then $\mathbf{c}; \text{refl}_{\mathbf{G}'}^{\sqsubseteq} = \mathbf{c}$ and $\text{refl}_{\mathbf{G}}^{\sqsubseteq}; \mathbf{c} = \mathbf{c}$.*

PROOF. We proceed by induction on $\mathbf{c} : \mathbf{G} \sqsubseteq \mathbf{G}'$.

Case ($\mathbf{B} : \mathbf{B} \sqsubseteq \mathbf{B}$). Since, $\text{refl}_{\mathbf{B}}^{\sqsubseteq} = \mathbf{B}$ and $\mathbf{B};\mathbf{B} = \mathbf{B}$, the result holds.

Case ($\mathbf{X} : \mathbf{X} \sqsubseteq \mathbf{X}$). Since, $\text{refl}_{\mathbf{X}}^{\sqsubseteq} = \mathbf{X}$ and $\mathbf{X};\mathbf{X} = \mathbf{X}$, the result holds.

Case ($\mathbf{c}_1 \rightarrow \mathbf{c}_2 : \mathbf{G}_{11} \rightarrow \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}$). We know that $\mathbf{c}_1 : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{c}_2 : \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$. By the inductive hypothesis, we get that $\text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq};\mathbf{c}_1 = \mathbf{c}_1$, $\mathbf{c}_1;\text{refl}_{\mathbf{G}_{21}}^{\sqsubseteq} = \mathbf{c}_1$, $\text{refl}_{\mathbf{G}_{12}}^{\sqsubseteq};\mathbf{c}_2 = \mathbf{c}_2$ and $\mathbf{c}_2;\text{refl}_{\mathbf{G}_{22}}^{\sqsubseteq} = \mathbf{c}_2$. We know that $\text{refl}_{\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}}^{\sqsubseteq} = \text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq} \rightarrow \text{refl}_{\mathbf{G}_{12}}^{\sqsubseteq}$ and $\text{refl}_{\mathbf{G}_{21} \rightarrow \mathbf{G}_{22}}^{\sqsubseteq} = \text{refl}_{\mathbf{G}_{21}}^{\sqsubseteq} \rightarrow \text{refl}_{\mathbf{G}_{22}}^{\sqsubseteq}$. We are required to prove that $\text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq} \rightarrow \text{refl}_{\mathbf{G}_{12}}^{\sqsubseteq};\mathbf{c}_1 \rightarrow \mathbf{c}_2 = \mathbf{c}_1 \rightarrow \mathbf{c}_2$ and $\mathbf{c}_1 \rightarrow \mathbf{c}_2;\text{refl}_{\mathbf{G}_{21}}^{\sqsubseteq} \rightarrow \text{refl}_{\mathbf{G}_{22}}^{\sqsubseteq} = \mathbf{c}_1 \rightarrow \mathbf{c}_2$. Or what is the same, we are required to show that $\text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq};\mathbf{c}_1 = \mathbf{c}_1$, $\mathbf{c}_1;\text{refl}_{\mathbf{G}_{21}}^{\sqsubseteq} = \mathbf{c}_1$, $\text{refl}_{\mathbf{G}_{12}}^{\sqsubseteq};\mathbf{c}_2 = \mathbf{c}_2$ and $\mathbf{c}_2;\text{refl}_{\mathbf{G}_{22}}^{\sqsubseteq} = \mathbf{c}_2$, as we already showed.

Case ($\mathbf{c}_1 \times \mathbf{c}_2 : \mathbf{G}_{11} \times \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{21} \times \mathbf{G}_{22}$). We know that $\mathbf{c}_1 : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{c}_2 : \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$. By the inductive hypothesis, we get that $\text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq};\mathbf{c}_1 = \mathbf{c}_1$, $\mathbf{c}_1;\text{refl}_{\mathbf{G}_{21}}^{\sqsubseteq} = \mathbf{c}_1$, $\text{refl}_{\mathbf{G}_{12}}^{\sqsubseteq};\mathbf{c}_2 = \mathbf{c}_2$ and $\mathbf{c}_2;\text{refl}_{\mathbf{G}_{22}}^{\sqsubseteq} = \mathbf{c}_2$. We know that $\text{refl}_{\mathbf{G}_{11} \times \mathbf{G}_{12}}^{\sqsubseteq} = \text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq} \times \text{refl}_{\mathbf{G}_{12}}^{\sqsubseteq}$ and $\text{refl}_{\mathbf{G}_{21} \times \mathbf{G}_{22}}^{\sqsubseteq} = \text{refl}_{\mathbf{G}_{21}}^{\sqsubseteq} \times \text{refl}_{\mathbf{G}_{22}}^{\sqsubseteq}$. We are required to prove that $\text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq} \times \text{refl}_{\mathbf{G}_{12}}^{\sqsubseteq};\mathbf{c}_1 \times \mathbf{c}_2 = \mathbf{c}_1 \times \mathbf{c}_2$ and $\mathbf{c}_1 \times \mathbf{c}_2;\text{refl}_{\mathbf{G}_{21}}^{\sqsubseteq} \times \text{refl}_{\mathbf{G}_{22}}^{\sqsubseteq} = \mathbf{c}_1 \times \mathbf{c}_2$. Or what is the same, we are required to show that $\text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq};\mathbf{c}_1 = \mathbf{c}_1$, $\mathbf{c}_1;\text{refl}_{\mathbf{G}_{21}}^{\sqsubseteq} = \mathbf{c}_1$, $\text{refl}_{\mathbf{G}_{12}}^{\sqsubseteq};\mathbf{c}_2 = \mathbf{c}_2$ and $\mathbf{c}_2;\text{refl}_{\mathbf{G}_{22}}^{\sqsubseteq} = \mathbf{c}_2$, as we already showed.

Case ($\forall \mathbf{X}.\mathbf{c}_1 : \forall \mathbf{X}.\mathbf{G}_{11} \sqsubseteq \forall \mathbf{X}.\mathbf{G}_{21}$). We know that $\mathbf{c}_1 : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$. By the inductive hypothesis, we get that $\text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq};\mathbf{c}_1 = \mathbf{c}_1$ and $\mathbf{c}_1;\text{refl}_{\mathbf{G}_{21}}^{\sqsubseteq} = \mathbf{c}_1$. We know that $\text{refl}_{\forall \mathbf{X}.\mathbf{G}_{11}}^{\sqsubseteq} = \forall \mathbf{X}.\text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq}$ and $\text{refl}_{\forall \mathbf{X}.\mathbf{G}_{21}}^{\sqsubseteq} = \forall \mathbf{X}.\text{refl}_{\mathbf{G}_{21}}^{\sqsubseteq}$. We are required to prove that $\forall \mathbf{X}.\text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq};\forall \mathbf{X}.\mathbf{c}_1 = \forall \mathbf{X}.\mathbf{c}_1$ and $\forall \mathbf{X}.\mathbf{c}_1;\forall \mathbf{X}.\text{refl}_{\mathbf{G}_{21}}^{\sqsubseteq} = \forall \mathbf{X}.\mathbf{c}_1$. Or what is the same, we are required to show that $\text{refl}_{\mathbf{G}_{11}}^{\sqsubseteq};\mathbf{c}_1 = \mathbf{c}_1$ and $\mathbf{c}_1;\text{refl}_{\mathbf{G}_{21}}^{\sqsubseteq} = \mathbf{c}_1$, as we already showed.

Case ($\text{inj}_{\mathbf{B}} : \mathbf{B} \sqsubseteq ?_{\delta}$). We know that $\text{refl}_{\mathbf{B}}^{\sqsubseteq} = \mathbf{B}$ and $\text{refl}_{?_{\delta}}^{\sqsubseteq} = \text{inj}_{?}$. Since, $\mathbf{B};\text{inj}_{\mathbf{B}} = \text{inj}_{\mathbf{B}}$ and $\text{inj}_{\mathbf{B}};\text{inj}_{?} = \text{inj}_{\mathbf{B}}$, the result holds.

Case ($\text{inj}_{\mathbf{X}} : \mathbf{F} \sqsubseteq ?_{\delta} \wedge \mathbf{X} : \mathbf{F} \in \delta$). We know that $\text{refl}_{\mathbf{F}}^{\sqsubseteq} = \mathbf{F}$ and $\text{refl}_{?_{\delta}}^{\sqsubseteq} = \text{inj}_{?}$. Since, $\mathbf{F};\text{inj}_{\mathbf{X}} = \text{inj}_{\mathbf{X}}$ and $\text{inj}_{\mathbf{X}};\text{inj}_{?} = \text{inj}_{\mathbf{X}}$, the result holds.

Case ($\text{inj}_{\rightarrow}(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_{\delta}$). We know that $\text{refl}_{?_{\delta}}^{\sqsubseteq} = \text{inj}_{?}$. Also, we know that $\mathbf{c}' : \mathbf{G} \sqsubseteq ?_{\delta} \rightarrow ?_{\delta}$. By the inductive hypothesis, we get that $\text{refl}_{\mathbf{G}}^{\sqsubseteq};\mathbf{c}' = \mathbf{c}'$. We are required to prove that $\text{inj}_{\rightarrow}(\mathbf{c}');\text{inj}_{?} = \text{inj}_{\rightarrow}(\mathbf{c}')$, which holds immediately. Also, we are required to prove that $\text{refl}_{\mathbf{G}}^{\sqsubseteq};\text{inj}_{\rightarrow}(\mathbf{c}') = \text{inj}_{\rightarrow}(\mathbf{c}')$. Thus, we are required to prove that $\text{refl}_{\mathbf{G}}^{\sqsubseteq};\mathbf{c}' = \mathbf{c}'$, as we already showed.

Case ($\text{inj}_{\times}(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_{\delta}$). We know that $\text{refl}_{?_{\delta}}^{\sqsubseteq} = \text{inj}_{?}$. Also, we know that $\mathbf{c}' : \mathbf{G} \sqsubseteq ?_{\delta} \times ?_{\delta}$. By the inductive hypothesis, we get that $\text{refl}_{\mathbf{G}}^{\sqsubseteq};\mathbf{c}' = \mathbf{c}'$. We are required to prove that $\text{inj}_{\times}(\mathbf{c}');\text{inj}_{?} = \text{inj}_{\times}(\mathbf{c}')$, which holds immediately. Also, we are required to prove that $\text{refl}_{\mathbf{G}}^{\sqsubseteq};\text{inj}_{\times}(\mathbf{c}') = \text{inj}_{\times}(\mathbf{c}')$. Thus, we are required to prove that $\text{refl}_{\mathbf{G}}^{\sqsubseteq};\mathbf{c}' = \mathbf{c}'$, as we already showed.

Case ($\text{inj}_{\vee}(\mathbf{c}') : \mathbf{G} \sqsubseteq ?_{\delta}$). We know that $\text{refl}_{?_{\delta}}^{\sqsubseteq} = \text{inj}_{?}$. Also, we know that $\mathbf{c}' : \mathbf{G} \sqsubseteq \forall \mathbf{X}.\mathbf{?}_{\delta}.\mathbf{X}$. By the inductive hypothesis, we get that $\text{refl}_{\mathbf{G}}^{\sqsubseteq};\mathbf{c}' = \mathbf{c}'$. We are required to prove that $\text{inj}_{\vee}(\mathbf{c}');\text{inj}_{?} = \text{inj}_{\vee}(\mathbf{c}')$, which holds immediately. Also, we are required to prove that $\text{refl}_{\mathbf{G}}^{\sqsubseteq};\text{inj}_{\vee}(\mathbf{c}') = \text{inj}_{\vee}(\mathbf{c}')$. Thus, we are required to prove that $\text{refl}_{\mathbf{G}}^{\sqsubseteq};\mathbf{c}' = \mathbf{c}'$, as we already showed.

Case ($\text{inj}_{?} : ?_{\delta} \sqsubseteq ?_{\delta'}$). The result holds immediately since $\text{refl}_{?_{\delta}}^{\sqsubseteq} = \text{inj}_{?}$, $\text{refl}_{?_{\delta'}}^{\sqsubseteq} = \text{inj}_{?}$ and $\text{inj}_{?};\text{inj}_{?} = \text{inj}_{?}$.

□

Lemma B.88 Suppose $\vdash \mathbf{c} : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$.

1. If $\mathbf{t}_1 = \mathbf{v}_1$, then $\mathbf{t}_2 = \mathbf{v}_2$.
2. If $\mathbf{t}_1 \mapsto \mathbf{t}'_1$, then $\mathbf{t}_2 \mapsto \mathbf{t}'_2$, and we have $\vdash \mathbf{c} : \mathbf{t}'_1 \sqsubseteq \mathbf{t}'_2$.

PROOF. We prove Case (1), directly, by Lemma B.80. We prove Case (2) by induction on reduction $\mathbf{t}_1 \mapsto \mathbf{t}'_1$.

Case $(\mathbf{t}_1 \mapsto \mathbf{t}'_1)$. By Lemma B.89, we know that $\mathbf{t}_2 \mapsto \mathbf{t}'_2$ and $\vdash \mathbf{c} : \mathbf{t}'_1 \sqsubseteq \mathbf{t}'_2$. Thus, the result holds immediately.

Case $(\varepsilon_{11} \mathbf{t}_{11} :: \mathbf{G}_{11} \mapsto \varepsilon_{11} \mathbf{t}'_{11} :: \mathbf{G}_{11})$. By inspection on $\vdash \mathbf{c} : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$ (\sqsubseteq_{asc} rule), we know that $\mathbf{t}_2 = \varepsilon_{22} \mathbf{t}_{22} :: \mathbf{G}_{22}$, $[\mathbf{c}] \varepsilon_{11} \sqsubseteq \varepsilon_{22} [\mathbf{c}]$, $\vdash \mathbf{c}' : \mathbf{t}_{11} \sqsubseteq \mathbf{t}_{22}$ and $\mathbf{c} : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{22}$. By induction hypothesis on $\mathbf{t}_{11} \mapsto \mathbf{t}'_{11}$, we get that $\mathbf{t}_{22} \mapsto \mathbf{t}'_{22}$ and $\vdash \mathbf{c}' : \mathbf{t}_{11} \sqsubseteq \mathbf{t}_{22}$. Then, by (\sqsubseteq_{asc}) rule, we know that $\vdash \mathbf{c} : \varepsilon_{11} \mathbf{t}'_{11} :: \mathbf{G}_{11} \sqsubseteq \varepsilon_{22} \mathbf{t}'_{22} :: \mathbf{G}_{22}$ and the result holds.

Case $(\mathbf{t}_{11} \mathbf{t}_{12} \mapsto \mathbf{t}'_{11} \mathbf{t}_{12})$. By inspection on $\vdash \mathbf{c} : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$ (\sqsubseteq_{app} rule), we know that $\mathbf{t}_2 = \mathbf{t}_{21} \mathbf{t}_{22}$, $\vdash \mathbf{c}' \rightarrow \mathbf{c} : \mathbf{t}_{11} \sqsubseteq \mathbf{t}_{21}$ and $\vdash \mathbf{c}' : \mathbf{t}_{12} \sqsubseteq \mathbf{t}_{22}$. By induction hypothesis on $\mathbf{t}_{11} \mapsto \mathbf{t}'_{11}$, we know that $\mathbf{t}_{21} \mapsto \mathbf{t}'_{21}$ and $\vdash \mathbf{c}' \rightarrow \mathbf{c} : \mathbf{t}'_{11} \sqsubseteq \mathbf{t}'_{21}$. Then, by (\sqsubseteq_{app}) rule, we know that $\vdash \mathbf{c} : \mathbf{t}'_{11} \mathbf{t}_{12} \sqsubseteq \mathbf{t}'_{21} \mathbf{t}_{22}$ and the result holds.

Case $(\mathbf{v}_{11} \mathbf{t}_{12} \mapsto \mathbf{v}_{11} \mathbf{t}'_{12})$. By inspection on $\vdash \mathbf{c} : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$ (\sqsubseteq_{app} rule) and Lemma B.80, we know that $\mathbf{t}_2 = \mathbf{v}_{21} \mathbf{t}_{22}$, $\vdash \mathbf{c}' \rightarrow \mathbf{c} : \mathbf{v}_{11} \sqsubseteq \mathbf{v}_{21}$ and $\vdash \mathbf{c}' : \mathbf{t}_{12} \sqsubseteq \mathbf{t}_{22}$. By induction hypothesis on $\mathbf{t}_{12} \mapsto \mathbf{t}'_{12}$, we know that $\mathbf{t}_{22} \mapsto \mathbf{t}'_{22}$ and $\vdash \mathbf{c}' : \mathbf{t}'_{12} \sqsubseteq \mathbf{t}'_{22}$. Then, by (\sqsubseteq_{app}) rule, we know that $\vdash \mathbf{c} : \mathbf{v}_{11} \mathbf{t}'_{12} \sqsubseteq \mathbf{v}_{21} \mathbf{t}'_{22}$ and the result holds.

Case $(\langle \mathbf{t}_{11}, \mathbf{t}_{12} \rangle \mapsto \langle \mathbf{t}'_{11}, \mathbf{t}_{12} \rangle)$. By inspection on $\vdash \mathbf{c} : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$ ($\sqsubseteq_{\text{pair}}$ rule), we know that $\mathbf{t}_2 = \langle \mathbf{t}_{21}, \mathbf{t}_{22} \rangle$, $\vdash \mathbf{c}_1 : \mathbf{t}_{11} \sqsubseteq \mathbf{t}_{21}$ and $\vdash \mathbf{c}_2 : \mathbf{t}_{12} \sqsubseteq \mathbf{t}_{22}$, where $\mathbf{c} = \mathbf{c}_1 \times \mathbf{c}_2$. By induction hypothesis on $\mathbf{t}_{11} \mapsto \mathbf{t}'_{11}$, we know that $\mathbf{t}_{21} \mapsto \mathbf{t}'_{21}$ and $\vdash \mathbf{c}_1 : \mathbf{t}'_{11} \sqsubseteq \mathbf{t}'_{21}$. Then, by ($\sqsubseteq_{\text{pair}}$) rule, we know that $\vdash \mathbf{c}_1 \times \mathbf{c}_2 : \langle \mathbf{t}'_{11}, \mathbf{t}_{12} \rangle \sqsubseteq \langle \mathbf{t}'_{21}, \mathbf{t}_{22} \rangle$ and the result holds.

Case $(\langle \mathbf{v}_{11}, \mathbf{t}_{12} \rangle \mapsto \langle \mathbf{v}_{11}, \mathbf{t}'_{12} \rangle)$. By inspection on $\vdash \mathbf{c} : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$ ($\sqsubseteq_{\text{pair}}$ rule) and Lemma B.80, we know that $\mathbf{t}_2 = \langle \mathbf{v}_{21}, \mathbf{t}_{22} \rangle$, $\vdash \mathbf{c}_1 : \mathbf{v}_{11} \sqsubseteq \mathbf{v}_{21}$ and $\vdash \mathbf{c}_2 : \mathbf{t}_{12} \sqsubseteq \mathbf{t}_{22}$, where $\mathbf{c} = \mathbf{c}_1 \times \mathbf{c}_2$. By induction hypothesis on $\mathbf{t}_{12} \mapsto \mathbf{t}'_{12}$, we know that $\mathbf{t}_{22} \mapsto \mathbf{t}'_{22}$ and $\vdash \mathbf{c}_2 : \mathbf{t}'_{12} \sqsubseteq \mathbf{t}'_{22}$. Then, by ($\sqsubseteq_{\text{pair}}$) rule, we know that $\vdash \mathbf{c}_1 \times \mathbf{c}_2 : \langle \mathbf{v}_{11}, \mathbf{t}'_{12} \rangle \sqsubseteq \langle \mathbf{v}_{21}, \mathbf{t}'_{22} \rangle$ and the result holds.

Case $(\mathbf{t}_{11} [\mathbf{F}] \mapsto \mathbf{t}'_{11} [\mathbf{F}])$. By inspection on $\vdash \mathbf{c} : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$ ($\sqsubseteq_{\text{appG}}$ rule), $\mathbf{t}_2 = \mathbf{t}_{22} [\mathbf{G}_{22}]$ and $\vdash \forall \mathbf{X}. \mathbf{c}' : \mathbf{t}_{11} \sqsubseteq \mathbf{t}_{22}$, where $\mathbf{c} = \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}]$. By induction hypothesis on $\mathbf{t}_{11} \mapsto \mathbf{t}'_{11}$, we know that $\mathbf{t}_{22} \mapsto \mathbf{t}'_{22}$ and $\vdash \forall \mathbf{X}. \mathbf{c}' : \mathbf{t}'_{11} \sqsubseteq \mathbf{t}'_{22}$. Then, by ($\sqsubseteq_{\text{appG}}$) rule, we know that $\vdash \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{t}'_{11} [\mathbf{F}] \sqsubseteq \mathbf{t}'_{22} [\mathbf{F}]$ and the result holds.

Case $(\pi_i(\mathbf{t}_{11}) \mapsto \pi_i(\mathbf{t}'_{11}))$. By inspection on $\vdash \mathbf{c} : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$ ($\sqsubseteq_{\text{proj}i}$ rule), $\mathbf{t}_2 = \pi_i(\mathbf{t}_{22})$ and $\vdash \mathbf{c}_1 \times \mathbf{c}_2 : \mathbf{t}_{11} \sqsubseteq \mathbf{t}_{22}$, where $\mathbf{c} = \mathbf{c}_i$. By induction hypothesis on $\mathbf{t}_{11} \mapsto \mathbf{t}'_{11}$, we know that $\mathbf{t}_{22} \mapsto \mathbf{t}'_{22}$ and $\vdash \mathbf{c}_1 \times \mathbf{c}_2 : \mathbf{t}'_{11} \sqsubseteq \mathbf{t}'_{22}$. Then, by ($\sqsubseteq_{\text{proj}i}$) rule, we know that $\vdash \mathbf{c}_i : \pi_i(\mathbf{t}'_{11}) \sqsubseteq \pi_i(\mathbf{t}'_{22})$ and the result holds.

□

Lemma B.89 If $\vdash \mathbf{c} : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$ and $\mathbf{t}_1 \mapsto \mathbf{t}'_1$, then $\mathbf{t}_2 \mapsto \mathbf{t}'_2$ and $\vdash \mathbf{c} : \mathbf{t}'_1 \sqsubseteq \mathbf{t}'_2$.

PROOF. We proceed by induction on $\vdash \mathbf{c} : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$.

Case $(\vdash \mathbf{c} : \mathbf{b} \sqsubseteq \mathbf{b})$. This case does not apply because \mathbf{b} is not a term \mathbf{t} ; therefore it can not reduce.

Case $(\vdash \mathbf{c} : (\lambda \mathbf{x} : \mathbf{G}_1^*. \mathbf{t}_1^*) \sqsubseteq (\lambda \mathbf{x} : \mathbf{G}_2^*. \mathbf{t}_2^*))$. This case does not apply because $(\lambda \mathbf{x} : \mathbf{G}_1^*. \mathbf{t}_1^*)$ is not a term \mathbf{t} ; therefore it can not reduce.

Case $(\vdash \mathbf{c} : (\Lambda \mathbf{X}. \mathbf{t}_1^*) \sqsubseteq (\Lambda \mathbf{X}. \mathbf{t}_2^*))$. This case does not apply because $(\Lambda \mathbf{X}. \mathbf{t}_1^*)$ is not a term \mathbf{t} ; therefore it can not reduce.

Case $(\vdash \mathbf{c} : \langle \mathbf{s}_{11}, \mathbf{s}_{12} \rangle \sqsubseteq \langle \mathbf{s}_{21}, \mathbf{s}_{22} \rangle)$. Similar to functions.

Case $(\vdash \mathbf{c} : \varepsilon_1 s_1^* :: \mathbf{G}_1 \sqsubseteq \varepsilon_2 s_2^* :: \mathbf{G}_2)$. We know by $(\sqsubseteq \text{asc})$ rule that

$$\frac{[\mathbf{c}']\varepsilon_1 \sqsubseteq \varepsilon_2[\mathbf{c}] \quad \vdash \mathbf{c}' : s_1^* \sqsubseteq s_2^* \quad \mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2}{\vdash \mathbf{c} : \varepsilon_1 s_1^* :: \mathbf{G}_1 \sqsubseteq \varepsilon_2 s_2^* :: \mathbf{G}_2}$$

Since $\mathbf{t}_1 \longrightarrow \mathbf{t}'_1$, we know that $\mathbf{t}_1 = \varepsilon_1(\varepsilon_{11} \mathbf{u}_{11} :: \mathbf{G}_1^*) :: \mathbf{G}_1$, where $s_1^* = \varepsilon_{11} \mathbf{u}_{11} :: \mathbf{G}_1^*$. Since $\vdash \mathbf{c}' : s_1^* \sqsubseteq s_2^*$ and s_1^* is a value, by Lemma B.80, we know that s_2^* is a value, and therefore, $\mathbf{t}_2 = \varepsilon_2(\varepsilon_{22} \mathbf{u}_{22} :: \mathbf{G}_2^*) :: \mathbf{G}_2$, where $s_2^* = \varepsilon_{22} \mathbf{u}_{22} :: \mathbf{G}_2^*$. By the reduction rule for ascriptions, we know that

$$\varepsilon_1(\varepsilon_{11} \mathbf{u}_{11} :: \mathbf{G}_1^*) :: \mathbf{G}_1 \longrightarrow (\varepsilon_{11} \mathbin{\text{\textcircled{;}}} \varepsilon_1) \mathbf{u}_{11} :: \mathbf{G}_1$$

Since $\vdash \mathbf{c}' : s_1^* \sqsubseteq s_2^*$, by $(\sqsubseteq \text{asc})$ rule, we know that

$$\frac{[\mathbf{c}']\varepsilon_{11} \sqsubseteq \varepsilon_{22}[\mathbf{c}'] \quad \vdash \mathbf{c}'' : \mathbf{u}_{11} \sqsubseteq \mathbf{u}_{22} \quad \mathbf{c}' : \mathbf{G}_1^* \sqsubseteq \mathbf{G}_2^*}{\vdash \mathbf{c}' : \varepsilon_{11} \mathbf{u}_{11} :: \mathbf{G}_1^* \sqsubseteq \varepsilon_{22} \mathbf{u}_{22} :: \mathbf{G}_2^*}$$

Since $[\mathbf{c}'']\varepsilon_{11} \sqsubseteq \varepsilon_{22}[\mathbf{c}']$, $[\mathbf{c}']\varepsilon_1 \sqsubseteq \varepsilon_2[\mathbf{c}]$ and $(\varepsilon_{11} \mathbin{\text{\textcircled{;}}} \varepsilon_1)$ is defined, by Lemma B.78, we get that $[\mathbf{c}''](\varepsilon_{11} \mathbin{\text{\textcircled{;}}} \varepsilon_1) \sqsubseteq (\varepsilon_{22} \mathbin{\text{\textcircled{;}}} \varepsilon_2)[\mathbf{c}]$. Therefore, we get that

$$\varepsilon_2(\varepsilon_{22} \mathbf{u}_{22} :: \mathbf{G}_2^*) :: \mathbf{G}_2 \longrightarrow (\varepsilon_{22} \mathbin{\text{\textcircled{;}}} \varepsilon_2) \mathbf{u}_{22} :: \mathbf{G}_2$$

Then, by $(\sqsubseteq \text{asc})$ rule, we get that

$$\frac{[\mathbf{c}''](\varepsilon_{11} \mathbin{\text{\textcircled{;}}} \varepsilon_1) \sqsubseteq (\varepsilon_{22} \mathbin{\text{\textcircled{;}}} \varepsilon_2)[\mathbf{c}] \quad \vdash \mathbf{c}'' : \mathbf{u}_{11} \sqsubseteq \mathbf{u}_{22} \quad \mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2}{\vdash \mathbf{c} : (\varepsilon_{11} \mathbin{\text{\textcircled{;}}} \varepsilon_1) \mathbf{u}_1 :: \mathbf{G}_1 \sqsubseteq (\varepsilon_{22} \mathbin{\text{\textcircled{;}}} \varepsilon_2) \mathbf{u}_2 :: \mathbf{G}_2}$$

Thus, the result holds.

Case $(\vdash \mathbf{c} : \mathbf{t}_{11}^* \mathbf{t}_{12}^* \sqsubseteq \mathbf{t}_{21}^* \mathbf{t}_{22}^*)$. We know that

$$\frac{\vdash \mathbf{c}' \rightarrow \mathbf{c} : \mathbf{t}_{11}^* \sqsubseteq \mathbf{t}_{21}^* \quad \vdash \mathbf{c}' : \mathbf{t}_{12}^* \sqsubseteq \mathbf{t}_{22}^*}{\vdash \mathbf{c} : \mathbf{t}_{11}^* \mathbf{t}_{12}^* \sqsubseteq \mathbf{t}_{21}^* \mathbf{t}_{22}^*}$$

Since $\mathbf{t}_1 \longrightarrow \mathbf{t}'_1$, we know that $\mathbf{t}_{11}^* = \varepsilon_{11}(\lambda \mathbf{x} : \mathbf{G}_1^*. \mathbf{t}_{11}) :: \mathbf{G}_{12} \longrightarrow \mathbf{G}_{11}$ and $\mathbf{t}_{12}^* = \mathbf{v}_{12} = \varepsilon_{12} \mathbf{u}_{12} :: \mathbf{G}_{12}$. By Lemma B.80, we know that $\mathbf{t}_{21}^* = \varepsilon_{21}(\lambda \mathbf{x} : \mathbf{G}_2^*. \mathbf{t}_{21}) :: \mathbf{G}_{22} \longrightarrow \mathbf{G}_{21}$ and $\mathbf{t}_{22}^* = \mathbf{v}_{22} = \varepsilon_{22} \mathbf{u}_{22} :: \mathbf{G}_{22}$. By the reduction rules, we know that

$$(\varepsilon_{11}(\lambda \mathbf{x} : \mathbf{G}_1^*. \mathbf{t}_{11}) :: \mathbf{G}_{12} \longrightarrow \mathbf{G}_{11}) (\varepsilon_{12} \mathbf{u}_{12} :: \mathbf{G}_{12}) \longrightarrow \text{cod}(\varepsilon_{11})(\mathbf{t}_{11}[\mathbf{x} / ((\varepsilon_{12} \mathbin{\text{\textcircled{;}}} \text{dom}(\varepsilon_{11})) \mathbf{u}_{12} :: \mathbf{G}_1^*)]) :: \mathbf{G}_{11}$$

Since $\vdash \mathbf{c}' \rightarrow \mathbf{c} : \mathbf{t}_{11}^* \sqsubseteq \mathbf{t}_{21}^*$, by $(\sqsubseteq \text{asc})$ rule and $(\sqsubseteq \lambda)$ rules, we know that

$$\frac{[\mathbf{c}' \rightarrow \mathbf{c}_2]\varepsilon_{11} \sqsubseteq \varepsilon_{21}[\mathbf{c}' \rightarrow \mathbf{c}] \quad \mathbf{x} : \mathbf{c}_1 : \mathbf{G}_1^* \sqsubseteq \mathbf{G}_2^* \vdash \mathbf{c}_2 : \mathbf{t}_{11} \sqsubseteq \mathbf{t}_{21} \quad \mathbf{c}' \rightarrow \mathbf{c} : \mathbf{G}_{12} \longrightarrow \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{22} \longrightarrow \mathbf{G}_{21}}{\vdash \mathbf{c} : \varepsilon_{11}(\lambda \mathbf{x} : \mathbf{G}_1^*. \mathbf{t}_{11}) :: \mathbf{G}_{12} \longrightarrow \mathbf{G}_{11} \sqsubseteq \varepsilon_{21}(\lambda \mathbf{x} : \mathbf{G}_2^*. \mathbf{t}_{21}) :: \mathbf{G}_{22} \longrightarrow \mathbf{G}_{21}}$$

Also, since $\vdash \mathbf{c}' : \mathbf{v}_{12} \sqsubseteq \mathbf{v}_{22}$, we know, by $(\sqsubseteq \text{asc})$ rule, that

$$\frac{[c'']\varepsilon_{12} \sqsubseteq \varepsilon_{22}[c'] \quad \vdash c'' : \mathbf{u}_{12} \sqsubseteq \mathbf{u}_{22} \quad c' : \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}}{\vdash c' : \varepsilon_{12} \mathbf{u}_{12} :: \mathbf{G}_{12} \sqsubseteq \varepsilon_{22} \mathbf{u}_{22} :: \mathbf{G}_{22}}$$

Since $[c_1 \rightarrow c_2]\varepsilon_{11} \sqsubseteq \varepsilon_{21}[c' \rightarrow c]$, by Lemma B.83, we know that $[c']\text{dom}(\varepsilon_{11}) \sqsubseteq \text{dom}(\varepsilon_{21})[c_1]$ and $[c_2]\text{cod}(\varepsilon_{11}) \sqsubseteq \text{cod}(\varepsilon_{21})[c]$. Since $[c'']\varepsilon_{12} \sqsubseteq \varepsilon_{22}[c']$ and $[c']\text{dom}(\varepsilon_{11}) \sqsubseteq \text{dom}(\varepsilon_{21})[c_1]$, by Lemma B.78, we know that $[c''](\varepsilon_{12} \mathbin{\text{\$}} \text{dom}(\varepsilon_{11})) \sqsubseteq (\varepsilon_{22} \mathbin{\text{\$}} \text{dom}(\varepsilon_{21}))[c_1]$.

Therefore, we know that

$$(\varepsilon_{21} (\lambda \mathbf{x} : \mathbf{G}_2^*. \mathbf{t}_{21}) :: \mathbf{G}_{22} \longrightarrow \mathbf{G}_{21}) (\varepsilon_{22} \mathbf{u}_{22} : \mathbf{G}_{22}) \longrightarrow \text{cod}(\varepsilon_{21}) (\mathbf{t}_{21} [((\varepsilon_{22} \mathbin{\text{\$}} \text{dom}(\varepsilon_{21})) \mathbf{u}_{21} :: \mathbf{G}_2^*) / \mathbf{x}] :: \mathbf{G}_{21})$$

Thus, by the $(\sqsubseteq \text{asc})$ rule, $[c''](\varepsilon_{12} \mathbin{\text{\$}} \text{dom}(\varepsilon_{11})) \sqsubseteq (\varepsilon_{22} \mathbin{\text{\$}} \text{dom}(\varepsilon_{21}))[c_1]$, $c_1 : \mathbf{G}_1^* \sqsubseteq \mathbf{G}_2^*$ and $\vdash c'' : \mathbf{u}_{12} \sqsubseteq \mathbf{u}_{22}$, we know that $\vdash c_1 : ((\varepsilon_{12} \mathbin{\text{\$}} \text{dom}(\varepsilon_{11})) \mathbf{u}_{11} :: \mathbf{G}_1^*) \sqsubseteq ((\varepsilon_{22} \mathbin{\text{\$}} \text{dom}(\varepsilon_{21})) \mathbf{u}_{21} :: \mathbf{G}_2^*)$.

Therefore, since $\mathbf{x} : c_1 : \mathbf{G}_1^* \sqsubseteq \mathbf{G}_2^* \vdash c_2 : \mathbf{t}_{11} \sqsubseteq \mathbf{t}_{21}$ and $\vdash c_1 : ((\varepsilon_{12} \mathbin{\text{\$}} \text{dom}(\varepsilon_{11})) \mathbf{u}_{11} :: \mathbf{G}_1^*) \sqsubseteq ((\varepsilon_{22} \mathbin{\text{\$}} \text{dom}(\varepsilon_{21})) \mathbf{u}_{21} :: \mathbf{G}_2^*)$, by Lemma B.90, we know that

$$\vdash c_2 : (\mathbf{t}_{11} [((\varepsilon_{12} \mathbin{\text{\$}} \text{dom}(\varepsilon_{11})) \mathbf{u}_{11} :: \mathbf{G}_1^*) / \mathbf{x}]) \sqsubseteq (\mathbf{t}_{21} [((\varepsilon_{22} \mathbin{\text{\$}} \text{dom}(\varepsilon_{21})) \mathbf{u}_{21} :: \mathbf{G}_2^*) / \mathbf{x}])$$

Finally, since $[c_2]\text{cod}(\varepsilon_{11}) \sqsubseteq \text{cod}(\varepsilon_{21})[c]$ and $c : \mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$, by $(\sqsubseteq \text{asc})$ rule, the result holds.

$$\vdash c : \text{cod}(\varepsilon_{11}) (\mathbf{t}_{11} [((\varepsilon_{12} \mathbin{\text{\$}} \text{dom}(\varepsilon_{11})) \mathbf{u}_{11} :: \mathbf{G}_1^*) / \mathbf{x}]) :: \mathbf{G}_{11} \sqsubseteq \text{cod}(\varepsilon_{21}) (\mathbf{t}_{21} [((\varepsilon_{22} \mathbin{\text{\$}} \text{dom}(\varepsilon_{21})) \mathbf{u}_{21} :: \mathbf{G}_2^*) / \mathbf{x}]) :: \mathbf{G}_{21}$$

Case $(\vdash c : \mathbf{t}_1^* [\mathbf{F}] \sqsubseteq \mathbf{t}_2^* [\mathbf{F}])$. We know that

$$\frac{\vdash \forall \mathbf{X}. c' : \mathbf{t}_1^* \sqsubseteq \mathbf{t}_2^*}{\vdash c' [\mathbf{X} \mapsto \mathbf{F}] : \mathbf{t}_1^* [\mathbf{F}] \sqsubseteq \mathbf{t}_2^* [\mathbf{F}]}$$

Since $\mathbf{t}_1 \longrightarrow \mathbf{t}'_1$, we know that $\mathbf{t}_1^* = \varepsilon_{11} \Lambda \mathbf{X}. \mathbf{t}_{11} :: \forall \mathbf{X}. \mathbf{G}_{11}$. By Lemma B.80, we know that $\mathbf{t}_2^* = \varepsilon_{22} \Lambda \mathbf{X}. \mathbf{t}_{22} :: \forall \mathbf{X}. \mathbf{G}_{22}$. Since $\vdash \forall \mathbf{X}. c' : \mathbf{t}_1^* \sqsubseteq \mathbf{t}_2^*$, by $(\sqsubseteq \text{asc})$ and $(\sqsubseteq \Lambda)$ rules, we know that

$$\frac{[\forall \mathbf{X}. c'']\varepsilon_{11} \sqsubseteq \varepsilon_{22}[\forall \mathbf{X}. c'] \quad \vdash c'' : \mathbf{t}_{11} \sqsubseteq \mathbf{t}_{22} \quad \forall \mathbf{X}. c' : \forall \mathbf{X}. \mathbf{G}_{11} \sqsubseteq \forall \mathbf{X}. \mathbf{G}_{22}}{\vdash \forall \mathbf{X}. c' : \varepsilon_{11} \Lambda \mathbf{X}. \mathbf{t}_{11} :: \forall \mathbf{X}. \mathbf{G}_{11} \sqsubseteq \varepsilon_{22} \Lambda \mathbf{X}. \mathbf{t}_{22} :: \forall \mathbf{X}. \mathbf{G}_{22}}$$

By the reduction rules, we know that

$$(\varepsilon_{11} (\Lambda \mathbf{X}. \mathbf{t}_{11}) :: \forall \mathbf{X}. \mathbf{G}_{11})[\mathbf{F}] \longrightarrow (\text{schm}(\varepsilon_{11}) \mathbf{t}_{11} :: \mathbf{G}_{11})[\mathbf{F}/\mathbf{X}]$$

We know that $(\text{schm}(\varepsilon_{11}) \mathbf{t}_{11} :: \mathbf{G}_{11})[\mathbf{F}/\mathbf{X}] = (\text{schm}(\varepsilon_{11})[\mathbf{X} \mapsto \mathbf{F}])(\mathbf{t}_{11}[\mathbf{X} \mapsto \mathbf{F}]) :: \mathbf{G}_{11}[\mathbf{F}/\mathbf{X}]$.

Also, by the reduction rules, we know that

$$(\varepsilon_{22} (\Lambda \mathbf{X}. \mathbf{t}_{22}) :: \forall \mathbf{X}. \mathbf{G}_{22})[\mathbf{F}] \longrightarrow (\text{schm}(\varepsilon_{22}) \mathbf{t}_{22} :: \mathbf{G}_{22})[\mathbf{F}/\mathbf{X}]$$

We know that $(\text{schm}(\varepsilon_{22}) \mathbf{t}_{22} :: \mathbf{G}_{22})[\mathbf{X} \mapsto \mathbf{F}] = (\text{schm}(\varepsilon_{22})[\mathbf{X} \mapsto \mathbf{F}])(\mathbf{t}_{22}[\mathbf{X} \mapsto \mathbf{F}]) :: \mathbf{G}_{22}[\mathbf{F}/\mathbf{X}]$. By Lemma 6.13, we know that $[c''[\mathbf{X} \mapsto \mathbf{F}]](\text{schm}(\varepsilon_{11})[\mathbf{X} \mapsto \mathbf{F}]) \sqsubseteq (\text{schm}(\varepsilon_{22})[\mathbf{X} \mapsto \mathbf{F}])[c'[\mathbf{X} \mapsto \mathbf{F}]]$. Since $\vdash c'' : \mathbf{t}_{11} \sqsubseteq \mathbf{t}_{22}$, by Lemma B.91, we know that $\Omega \vdash c''[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{t}_{11}[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{t}_{22}[\mathbf{X} \mapsto \mathbf{F}]$. By Lemma B.81, we know that $c'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}_{11}[\mathbf{F}/\mathbf{X}] \sqsubseteq \mathbf{G}_{22}[\mathbf{F}/\mathbf{X}]$.

Finally, by $(\sqsubseteq \text{asc})$ rule the result holds.

$$\vdash c'[\mathbf{X} \mapsto \mathbf{F}] : (\text{schm}(\varepsilon_{11}) \mathbf{t}_{11} :: \mathbf{G}_{11})[\mathbf{F}/\mathbf{X}] \sqsubseteq (\text{schm}(\varepsilon_{22}) \mathbf{t}_{22} :: \mathbf{G}_{22})[\mathbf{F}/\mathbf{X}]$$

Case ($\vdash \mathbf{c} : \pi_i(\mathbf{t}_1^*) \sqsubseteq \pi_i(\mathbf{t}_2^*)$). We know that

$$\frac{\vdash \mathbf{c}_1 \times \mathbf{c}_2 : \mathbf{t}_1^* \sqsubseteq \mathbf{t}_2^*}{\vdash \mathbf{c}_i : \pi_i(\mathbf{t}_1^*) \sqsubseteq \pi_i(\mathbf{t}_2^*)}$$

Since $\mathbf{t}_1 \longrightarrow \mathbf{t}'_1$, we know that $\mathbf{t}_1^* = \mathbf{v}_1 = \varepsilon_1 \langle \mathbf{u}_{11}, \mathbf{u}_{12} \rangle :: \mathbf{G}_{11} \times \mathbf{G}_{12}$.

By Lemma B.80, we know that $\mathbf{t}_2^* = \mathbf{v}_2 = \varepsilon_2 \langle \mathbf{u}_{21}, \mathbf{u}_{22} \rangle :: \mathbf{G}_{21} \times \mathbf{G}_{22}$. By the reduction rules, we know that

$$\pi_i(\mathbf{t}_1^*) \longrightarrow \pi_i(\varepsilon_1) \mathbf{u}_{1i} :: \mathbf{G}_{1i}$$

$$\pi_i(\mathbf{t}_2^*) \longrightarrow \pi_i(\varepsilon_2) \mathbf{u}_{2i} :: \mathbf{G}_{2i}$$

Since $\vdash \mathbf{c}_1 \times \mathbf{c}_2 : \mathbf{t}_1^* \sqsubseteq \mathbf{t}_2^*$, by (\sqsubseteq asc) rule and (\sqsubseteq pair) rules, we know that

$$\frac{[\mathbf{c}_1 \times \mathbf{c}'_2] \varepsilon_1 \sqsubseteq \varepsilon_2 [\mathbf{c}_1 \times \mathbf{c}_2] \quad \Omega \vdash \mathbf{c}_1 : \mathbf{u}_{11} \sqsubseteq \mathbf{u}_{21} \quad \Omega \vdash \mathbf{c}'_2 : \mathbf{u}_{12} \sqsubseteq \mathbf{u}_{22} \quad \mathbf{c}_1 \times \mathbf{c}_2 : \mathbf{G}_{11} \times \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{21} \times \mathbf{G}_{22}}{\vdash \mathbf{c}_1 \times \mathbf{c}_2 : \varepsilon_1 \langle \mathbf{u}_{11}, \mathbf{u}_{12} \rangle :: \mathbf{G}_{11} \times \mathbf{G}_{12} \sqsubseteq \varepsilon_2 \langle \mathbf{u}_{21}, \mathbf{u}_{22} \rangle :: \mathbf{G}_{21} \times \mathbf{G}_{22}}$$

Since $[\mathbf{c}_1 \times \mathbf{c}'_2] \varepsilon_1 \sqsubseteq \varepsilon_2 [\mathbf{c}_1 \times \mathbf{c}_2]$, by Lemma B.83, we know that $[\mathbf{c}'_i] \pi_i(\varepsilon_1) \sqsubseteq \pi_i(\varepsilon_2) [\mathbf{c}_i]$. Finally, since $[\mathbf{c}'_i] \pi_i(\varepsilon_1) \sqsubseteq \pi_i(\varepsilon_2) [\mathbf{c}_i]$, $\mathbf{c}_i : \mathbf{G}_{1i} \sqsubseteq \mathbf{G}_{2i}$ and $\Omega \vdash \mathbf{c}'_i : \mathbf{u}_{1i} \sqsubseteq \mathbf{u}_{2i}$, by (\sqsubseteq asc) rule, the result holds.

$$\vdash \mathbf{c}_i : \pi_i(\varepsilon_1) \mathbf{u}_{1i} :: \mathbf{G}_{1i} \sqsubseteq \pi_i(\varepsilon_2) \mathbf{u}_{2i} :: \mathbf{G}_{2i}$$

Case ($\vdash \mathbf{c} : \langle \mathbf{s}_{11}, \mathbf{s}_{12} \rangle \sqsubseteq \langle \mathbf{s}_{21}, \mathbf{s}_{22} \rangle$). We know that

$$\frac{\vdash \mathbf{c}_1 : \mathbf{s}_{11} \sqsubseteq \mathbf{s}_{21} \quad \vdash \mathbf{c}_2 : \mathbf{s}_{12} \sqsubseteq \mathbf{s}_{22}}{\vdash \mathbf{c}_1 \times \mathbf{c}_2 : \langle \mathbf{s}_{11}, \mathbf{s}_{12} \rangle \sqsubseteq \langle \mathbf{s}_{21}, \mathbf{s}_{22} \rangle}$$

Since $\mathbf{t}_1 \longrightarrow \mathbf{t}'_1$, we know that $\mathbf{s}_{1i} = v_{1i} = \varepsilon_{1i} \mathbf{u}_{1i} :: \mathbf{G}_{1i}$. Since $\vdash \mathbf{c}_1 : \mathbf{s}_{11} \sqsubseteq \mathbf{s}_{21}$ and $\vdash \mathbf{c}_2 : \mathbf{s}_{12} \sqsubseteq \mathbf{s}_{22}$, by Lemma B.80, we know that $\mathbf{s}_{2i} = v_{2i} = \varepsilon_{2i} \mathbf{u}_{2i} :: \mathbf{G}_{2i}$. By the reduction rules, we know that

$$\langle \mathbf{s}_{11}, \mathbf{s}_{12} \rangle \longrightarrow (\varepsilon_{11} \times \varepsilon_{12}) \langle \mathbf{u}_{11}, \mathbf{u}_{12} \rangle :: \mathbf{G}_{11} \times \mathbf{G}_{12}$$

$$\langle \mathbf{s}_{21}, \mathbf{s}_{22} \rangle \longrightarrow (\varepsilon_{21} \times \varepsilon_{22}) \langle \mathbf{u}_{21}, \mathbf{u}_{22} \rangle :: \mathbf{G}_{21} \times \mathbf{G}_{22}$$

Since $\vdash \mathbf{c}_1 : \mathbf{s}_{11} \sqsubseteq \mathbf{s}_{21}$ and $\vdash \mathbf{c}_2 : \mathbf{s}_{12} \sqsubseteq \mathbf{s}_{22}$, By (\sqsubseteq asc) rule, we know that

$$\frac{[\mathbf{c}'_i] \varepsilon_{1i} \sqsubseteq \varepsilon_{2i} [\mathbf{c}_i] \quad \vdash \mathbf{c}'_i : \mathbf{u}_{1i} \sqsubseteq \mathbf{u}_{2i} \quad \mathbf{c}_i : \mathbf{G}_{1i} \sqsubseteq \mathbf{G}_{2i}}{\vdash \mathbf{c}_i : \mathbf{s}_{1i} \sqsubseteq \mathbf{s}_{2i}}$$

Since $[\mathbf{c}_1] \varepsilon_{11} \sqsubseteq \varepsilon_{21} [\mathbf{c}_1]$ and $[\mathbf{c}'_2] \varepsilon_{12} \sqsubseteq \varepsilon_{22} [\mathbf{c}_2]$, by Lemma B.75, we get that $[\mathbf{c}_1 \times \mathbf{c}'_2] (\varepsilon_{11} \times \varepsilon_{12}) \sqsubseteq (\varepsilon_{21} \times \varepsilon_{22}) [\mathbf{c}_1 \times \mathbf{c}_2]$. Finally, since $\Omega \vdash \mathbf{c}'_i : \mathbf{u}_{1i} \sqsubseteq \mathbf{u}_{2i}$, $[\mathbf{c}_1 \times \mathbf{c}'_2] (\varepsilon_{11} \times \varepsilon_{12}) \sqsubseteq (\varepsilon_{21} \times \varepsilon_{22}) [\mathbf{c}_1 \times \mathbf{c}_2]$ and $\mathbf{c}_1 \times \mathbf{c}_2 : \mathbf{G}_{11} \times \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{21} \times \mathbf{G}_{22}$, by (\sqsubseteq asc) and (\sqsubseteq pair) rules, the result holds.

$$\vdash \mathbf{c}_1 \times \mathbf{c}_2 : (\varepsilon_{11} \times \varepsilon_{12}) \langle \mathbf{u}_{11}, \mathbf{u}_{12} \rangle :: \mathbf{G}_{11} \times \mathbf{G}_{12} \sqsubseteq (\varepsilon_{21} \times \varepsilon_{22}) \langle \mathbf{u}_{21}, \mathbf{u}_{22} \rangle :: \mathbf{G}_{21} \times \mathbf{G}_{22}$$

□

Lemma B.90 *If $\Omega, \mathbf{x} : \mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2 \vdash \mathbf{c}' : \mathbf{s}_1 \sqsubseteq \mathbf{s}_2$ and $\Omega \vdash \mathbf{c} : \mathbf{v}_1 \sqsubseteq \mathbf{v}_2$, then $\Omega \vdash \mathbf{c}' : \mathbf{s}_1[\mathbf{v}_1/\mathbf{x}] \sqsubseteq \mathbf{s}_2[\mathbf{v}_2/\mathbf{x}]$.*

PROOF. We proceed by induction on $\Omega, \mathbf{x} : \mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2 \vdash \mathbf{c}' : \mathbf{s}_1 \sqsubseteq \mathbf{s}_2$. Let $\Omega' = \Omega, \mathbf{x} : \mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$.

Case ($\sqsubseteq b$). We know that

$$\sqsubseteq b \frac{}{\Omega' \vdash \mathbf{B} : \mathbf{b} \sqsubseteq \mathbf{b}}$$

Since $b[\mathbf{v}_1/\mathbf{x}] = b[\mathbf{v}_1/\mathbf{x}] = b$, the result follows immediately.

Case ($\sqsubseteq x$). We know that

$$\sqsubseteq x \frac{\mathbf{y} : \mathbf{c}'' : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2 \in \Omega'}{\Omega' \vdash \mathbf{c}'' : \mathbf{y} \sqsubseteq \mathbf{y}}$$

We have the following cases.

- $\mathbf{y} = \mathbf{x}$. The result follows immediately due to $\Omega \vdash \mathbf{c} : \mathbf{v}_1 \sqsubseteq \mathbf{v}_2$, $\mathbf{t}_1[\mathbf{v}_1/\mathbf{x}] = \mathbf{x}[\mathbf{v}_1/\mathbf{x}] = \mathbf{v}_1$ and $\mathbf{t}_2[\mathbf{v}_2/\mathbf{x}] = \mathbf{x}[\mathbf{v}_2/\mathbf{x}] = \mathbf{v}_2$.
- $\mathbf{y} \neq \mathbf{x}$. Since $\mathbf{y}[\mathbf{v}_1/\mathbf{x}] = \mathbf{y}[\mathbf{v}_1/\mathbf{x}] = \mathbf{y}$, the result follows immediately.

Case ($\sqsubseteq \lambda$). We know that

$$\sqsubseteq \lambda \frac{\Omega', \mathbf{y} : \mathbf{c}_1 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2 \vdash \mathbf{c}_2 : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2}{\Omega' \vdash \mathbf{c}_1 \rightarrow \mathbf{c}_2 : (\lambda \mathbf{y} : \mathbf{G}'_1. \mathbf{t}_1) \sqsubseteq (\lambda \mathbf{y} : \mathbf{G}'_2. \mathbf{t}_2)}$$

Note that we are required to prove that

$$\Omega \vdash \mathbf{c}_1 \rightarrow \mathbf{c}_2 : \lambda \mathbf{y} : \mathbf{G}'_1. (\mathbf{t}_1[\mathbf{v}_1/\mathbf{x}]) \sqsubseteq \lambda \mathbf{y} : \mathbf{G}'_2. (\mathbf{t}_2[\mathbf{v}_2/\mathbf{x}])$$

Or what is the same, we are required to prove that $\Omega, \mathbf{y} : \mathbf{c}_1 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2 \vdash \mathbf{c}_2 : \mathbf{t}_1[\mathbf{v}_1/\mathbf{x}] \sqsubseteq \mathbf{t}_2[\mathbf{v}_2/\mathbf{x}]$. The result follows immediately by the induction hypothesis on $\Omega', \mathbf{y} : \mathbf{c}_1 : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2 \vdash \mathbf{c}_2 : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$.

Case ($\sqsubseteq \Lambda$). We know that

$$\sqsubseteq \Lambda \frac{\Omega' \vdash \mathbf{c}_1 : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2}{\Omega' \vdash \forall \mathbf{X}. \mathbf{c}_1 : (\Lambda \mathbf{X}. \mathbf{t}_1) \sqsubseteq (\Lambda \mathbf{X}. \mathbf{t}_2)}$$

Note that we are required to prove that

$$\Omega \vdash \forall \mathbf{X}. \mathbf{c}_1 : \Lambda \mathbf{X}. (\mathbf{t}_1[\mathbf{v}_1/\mathbf{x}]) \sqsubseteq \Lambda \mathbf{X}. (\mathbf{t}_2[\mathbf{v}_2/\mathbf{x}])$$

Or what is the same, we are required to prove that $\Omega \vdash \mathbf{c}_1 : \mathbf{t}_1[\mathbf{v}_1/\mathbf{x}] \sqsubseteq \mathbf{t}_2[\mathbf{v}_2/\mathbf{x}]$. The result follows immediately by the induction hypothesis on $\Omega' \vdash \mathbf{c}_1 : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$.

Case ($\sqsubseteq \text{pair}$). We know that

$$\sqsubseteq \text{pair} \frac{\Omega' \vdash \mathbf{c}_1 : \mathbf{s}'_1 \sqsubseteq \mathbf{s}'_2 \quad \Omega' \vdash \mathbf{c}_2 : \mathbf{s}''_1 \sqsubseteq \mathbf{s}''_2}{\Omega' \vdash \mathbf{c}_1 \times \mathbf{c}_2 : \langle \mathbf{s}'_1, \mathbf{s}''_1 \rangle \sqsubseteq \langle \mathbf{s}'_2, \mathbf{s}''_2 \rangle}$$

Note that we are required to prove that

$$\Omega \vdash \mathbf{c}_1 \times \mathbf{c}_2 : \langle \mathbf{s}'_1[\mathbf{v}_1/\mathbf{x}], \mathbf{s}''_2[\mathbf{v}_1/\mathbf{x}] \rangle \sqsubseteq \langle \mathbf{s}'_1[\mathbf{v}_2/\mathbf{x}], \mathbf{s}''_2[\mathbf{v}_2/\mathbf{x}] \rangle$$

Or what is the same, we are required to prove that $\Omega \vdash \mathbf{c}_1 : \mathbf{s}'_1[\mathbf{v}_1/\mathbf{x}] \sqsubseteq \mathbf{s}'_1[\mathbf{v}_2/\mathbf{x}]$ and $\Omega \vdash \mathbf{c}_2 : \mathbf{s}''_2[\mathbf{v}_1/\mathbf{x}] \sqsubseteq \mathbf{s}''_2[\mathbf{v}_2/\mathbf{x}]$. The result follows immediately by the induction hypothesis on $\Omega' \vdash \mathbf{c}_1 : \mathbf{s}'_1 \sqsubseteq \mathbf{s}'_2$ and $\Omega' \vdash \mathbf{c}_2 : \mathbf{s}''_1 \sqsubseteq \mathbf{s}''_2$.

Case (\sqsubseteq_{app}). We know that

$$\sqsubseteq_{\text{app}} \frac{\Omega' \vdash \mathbf{c}_1 \longrightarrow \mathbf{c}_2 : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2 \quad \Omega' \vdash \mathbf{c}_1 : \mathbf{t}'_1 \sqsubseteq \mathbf{t}'_2}{\Omega' \vdash \mathbf{c}_2 : \mathbf{t}_1 \mathbf{t}'_1 \sqsubseteq \mathbf{t}_2 \mathbf{t}'_2}$$

Note that we are required to prove that

$$\Omega \vdash \mathbf{c}_2 : (\mathbf{t}_1[\mathbf{v}_1/\mathbf{x}]) (\mathbf{t}'_1[\mathbf{v}_1/\mathbf{x}]) \sqsubseteq (\mathbf{t}_2[\mathbf{v}_2/\mathbf{x}]) (\mathbf{t}'_2[\mathbf{v}_2/\mathbf{x}])$$

Or what is the same, we are required to prove that $\Omega \vdash \mathbf{c}_1 \longrightarrow \mathbf{c}_2 : \mathbf{t}_1[\mathbf{v}_1/\mathbf{x}] \sqsubseteq \mathbf{t}_2[\mathbf{v}_2/\mathbf{x}]$ and $\Omega \vdash \mathbf{c}_2 : \mathbf{t}'_1[\mathbf{v}_1/\mathbf{x}] \sqsubseteq \mathbf{t}'_2[\mathbf{v}_2/\mathbf{x}]$. The result follows immediately by the induction hypothesis on $\Omega' \vdash \mathbf{c}_1 \longrightarrow \mathbf{c}_2 : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$ and $\Omega' \vdash \mathbf{c}_1 : \mathbf{t}'_1 \sqsubseteq \mathbf{t}'_2$.

Case ($\sqsubseteq_{\text{pair}i}$). We know that

$$\sqsubseteq_{\text{pair}i} \frac{\Omega' \vdash \mathbf{c}_1 \times \mathbf{c}_2 : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2}{\Omega' \vdash \mathbf{c}_i : \pi_i(\mathbf{t}_1) \sqsubseteq \pi_i(\mathbf{t}_2)}$$

Note that we are required to prove that

$$\Omega \vdash \mathbf{c}_i : \pi_i(\mathbf{t}_1[\mathbf{v}_1/\mathbf{x}]) \sqsubseteq \pi_i(\mathbf{t}_2[\mathbf{v}_2/\mathbf{x}])$$

Or what is the same, we are required to prove that $\Omega \vdash \mathbf{c}_1 \times \mathbf{c}_2 : \mathbf{t}_1[\mathbf{v}_1/\mathbf{x}] \sqsubseteq \mathbf{t}_2[\mathbf{v}_2/\mathbf{x}]$. The result follows immediately by the induction hypothesis on $\Omega' \vdash \mathbf{c}_1 \times \mathbf{c}_2 : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$.

Case ($\sqsubseteq_{\text{appG}}$). We know that

$$\sqsubseteq_{\text{appG}} \frac{\Omega' \vdash \forall \mathbf{X}. \mathbf{c}_1 : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2}{\Omega' \vdash \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{t}_1[\mathbf{F}] \sqsubseteq \mathbf{t}_2[\mathbf{F}]}$$

Note that we are required to prove that

$$\Omega \vdash \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] : (\mathbf{t}_1[\mathbf{v}_1/\mathbf{x}][\mathbf{F}]) \sqsubseteq (\mathbf{t}_2[\mathbf{v}_2/\mathbf{x}][\mathbf{F}])$$

Or what is the same, we are required to prove that $\Omega \vdash \forall \mathbf{X}. \mathbf{c}_1 : \mathbf{t}_1[\mathbf{v}_1/\mathbf{x}] \sqsubseteq \mathbf{t}_2[\mathbf{v}_2/\mathbf{x}]$. The result follows immediately by the induction hypothesis on $\Omega' \vdash \forall \mathbf{X}. \mathbf{c}_1 : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$.

Case (\sqsubseteq_{asc}).

$$\sqsubseteq_{\text{asc}} \frac{[\mathbf{c}_1]\varepsilon_1 \sqsubseteq \varepsilon_2[\mathbf{c}_2] \quad \Omega' \vdash \mathbf{c}_1 : \mathbf{s}'_1 \sqsubseteq \mathbf{s}'_2 \quad \mathbf{c}_2 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2}{\Omega' \vdash \mathbf{c}_2 : \varepsilon_1 \mathbf{s}'_1 :: \mathbf{G}_1 \sqsubseteq \varepsilon_2 \mathbf{s}'_2 :: \mathbf{G}_2}$$

Note that we are required to prove that

$$\Omega \vdash \mathbf{c}_2 : (\varepsilon_1 \mathbf{s}'_1[\mathbf{v}_1/\mathbf{x}] :: \mathbf{G}_1) \sqsubseteq (\varepsilon_2 \mathbf{s}'_2[\mathbf{v}_2/\mathbf{x}] :: \mathbf{G}_2)$$

Or what is the same, we are required to prove that $\Omega \vdash \mathbf{c}_1 : \mathbf{s}'_1[\mathbf{v}_1/\mathbf{x}] \sqsubseteq \mathbf{s}'_2[\mathbf{v}_2/\mathbf{x}]$. The result follows immediately by the induction hypothesis on $\Omega' \vdash \mathbf{c}_1 : \mathbf{s}'_1 \sqsubseteq \mathbf{s}'_2$.

□

Lemma B.91 *If $\Omega \vdash \mathbf{c} : \mathbf{s} \sqsubseteq \mathbf{s}'$, then $\Omega[\mathbf{X} \mapsto \mathbf{F}] \vdash \mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] : (\mathbf{s}[\mathbf{X} \mapsto \mathbf{F}]) \sqsubseteq (\mathbf{s}'[\mathbf{X} \mapsto \mathbf{F}])$.*

PROOF. We proceed by induction on $\Omega \vdash \mathbf{c} : \mathbf{s} \sqsubseteq \mathbf{s}'$.

Case ($\Omega \vdash \mathbf{B} : \mathbf{b} \sqsubseteq \mathbf{b}$). We know that $\mathbf{B}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{B}$ and $\mathbf{b}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{b}$. The result holds immediate since $\Omega[\mathbf{X} \mapsto \mathbf{F}] \vdash \mathbf{B} : \mathbf{b} \sqsubseteq \mathbf{b}$, by rule $\sqsubseteq_{\mathbf{b}}$.

Case $(\Omega \vdash \mathbf{c} : \mathbf{x} \sqsubseteq \mathbf{x})$. Since $\mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, by Lemma B.81, we get that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}]$. We know that $\mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}] \in \Omega[\mathbf{X} \mapsto \mathbf{F}]$ and $\mathbf{x}[\mathbf{X} \mapsto \mathbf{F}] = \mathbf{x}$. The result holds immediate since $\Omega[\mathbf{X} \mapsto \mathbf{F}] \vdash \mathbf{c}[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{x} \sqsubseteq \mathbf{x}$, by rule $\sqsubseteq_{\mathbf{x}}$.

Case $(\Omega \vdash \mathbf{c}' \rightarrow \mathbf{c}'' : (\lambda \mathbf{x} : \mathbf{G}'_1. \mathbf{t}_1) \sqsubseteq (\lambda \mathbf{x} : \mathbf{G}'_2. \mathbf{t}_2))$. We know that $\Omega, \mathbf{x} : \mathbf{c}' : \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2 \vdash \mathbf{c}'' : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$. Therefore, by the induction hypothesis, we get that

$$\Omega[\mathbf{X} \mapsto \mathbf{F}], \mathbf{x} : \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}'_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}'_2[\mathbf{X} \mapsto \mathbf{F}] \vdash \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{t}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{t}_2[\mathbf{X} \mapsto \mathbf{F}]$$

as we are required to show, by rule \sqsubseteq_{λ} .

Case $(\Omega \vdash \forall \mathbf{X}. \mathbf{c}' : (\lambda \mathbf{X}. \mathbf{t}_1) \sqsubseteq (\lambda \mathbf{X}. \mathbf{t}_2))$. We know that $\Omega \vdash \mathbf{c}' : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$. Therefore, by the induction hypothesis, we get that

$$\Omega[\mathbf{X} \mapsto \mathbf{F}] \vdash \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{t}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{t}_2[\mathbf{X} \mapsto \mathbf{F}]$$

as we are required to show, by rule \sqsubseteq_{Λ} .

Case $(\Omega \vdash \mathbf{c}' : \varepsilon_1 \mathbf{s}_1 :: \mathbf{G}_1 \sqsubseteq \varepsilon_2 \mathbf{s}_2 :: \mathbf{G}_2)$. We know that $\Omega \vdash \mathbf{c}'' : \mathbf{s}_1 \sqsubseteq \mathbf{s}_2$. Therefore, by the induction hypothesis, we get that $\Omega[\mathbf{X} \mapsto \mathbf{F}] \vdash \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{s}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{s}_2[\mathbf{X} \mapsto \mathbf{F}]$. By Lemma B.81, we know that $\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{G}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{G}_2[\mathbf{X} \mapsto \mathbf{F}]$. By Lemma B.82, we know that $[\mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}]]_{\varepsilon_1}[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \varepsilon_2[\mathbf{X} \mapsto \mathbf{F}][\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}]]$. Therefore, the result holds by \sqsubseteq_{asc} rule.

Case $(\Omega \vdash \mathbf{c}' : \mathbf{t}_1 \mathbf{t}'_1 \sqsubseteq \mathbf{t}_2 \mathbf{t}'_2)$. We know that $\Omega \vdash \mathbf{c}'' \rightarrow \mathbf{c}' : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$ and $\Omega \vdash \mathbf{c}'' : \mathbf{t}'_1 \sqsubseteq \mathbf{t}'_2$. Therefore, by the induction hypothesis, we get that $\Omega[\mathbf{X} \mapsto \mathbf{F}] \vdash \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] \rightarrow \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{t}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{t}_2[\mathbf{X} \mapsto \mathbf{F}]$ and $\Omega[\mathbf{X} \mapsto \mathbf{F}] \vdash \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{t}'_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{t}'_2[\mathbf{X} \mapsto \mathbf{F}]$. Therefore, the result holds by \sqsubseteq_{app} rule.

Case $(\Omega \vdash \mathbf{c}' \times \mathbf{c}'' : \langle \mathbf{s}_1, \mathbf{s}'_1 \rangle \sqsubseteq \langle \mathbf{s}_2, \mathbf{s}'_2 \rangle)$. We know that $\Omega \vdash \mathbf{c}' : \mathbf{s}_1 \sqsubseteq \mathbf{s}_2$ and $\Omega \vdash \mathbf{c}'' : \mathbf{s}'_1 \sqsubseteq \mathbf{s}'_2$. Therefore, by the induction hypothesis, we get that $\Omega[\mathbf{X} \mapsto \mathbf{F}] \vdash \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{s}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{s}_2[\mathbf{X} \mapsto \mathbf{F}]$ and $\Omega[\mathbf{X} \mapsto \mathbf{F}] \vdash \mathbf{c}''[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{s}'_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{s}'_2[\mathbf{X} \mapsto \mathbf{F}]$. Therefore, the result holds by $\sqsubseteq_{\text{pair}}$ rule.

Case $(\Omega \vdash \mathbf{c}_i : \pi_i(\mathbf{t}_1) \sqsubseteq \pi_i(\mathbf{t}_2))$. We know that $\Omega \vdash \mathbf{c}_1 \times \mathbf{c}_2 : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$. Therefore, by the induction hypothesis, we get that $\Omega[\mathbf{X} \mapsto \mathbf{F}] \vdash \mathbf{c}_1[\mathbf{X} \mapsto \mathbf{F}] \times \mathbf{c}_2[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{t}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{t}_2[\mathbf{X} \mapsto \mathbf{F}]$. Therefore, the result holds by $\sqsubseteq_{\text{pair}i}$ rule.

Case $(\Omega \vdash \mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}] : \mathbf{t}_1[\mathbf{F}'] \sqsubseteq \mathbf{t}_2[\mathbf{F}'])$. We know that $\Omega \vdash \forall \mathbf{Y}. \mathbf{c}' : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$. Therefore, by the induction hypothesis, we get that $\Omega[\mathbf{X} \mapsto \mathbf{F}] \vdash \forall \mathbf{Y}. (\mathbf{c}'[\mathbf{X} \mapsto \mathbf{F}]) : \mathbf{t}_1[\mathbf{X} \mapsto \mathbf{F}] \sqsubseteq \mathbf{t}_2[\mathbf{X} \mapsto \mathbf{F}]$. Therefore, the result holds by $\sqsubseteq_{\text{appG}}$ rule.

□

B.5 The Static Language System \mathbf{F}_1

B.6 The Gradual Source Language $\mathbf{F}^?$

B.6.1 Lifting the Static Semantics

$$\text{ftv}(\cdot) = \cdot$$

$$\text{ftv}(\delta, \mathbf{X} : \mathbf{X}) = \text{ftv}(\delta) \cup \text{ftv}(\mathbf{X})$$

$\Delta; \Gamma \vdash t : T$ **Term typing**

$$\begin{array}{c}
(T_x) \frac{x : T \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x : T} \quad (T_b) \frac{\theta(b) = B \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash b : B} \quad (T_\lambda) \frac{\Delta; \Gamma, x : T_1 \vdash t : T_2}{\Delta; \Gamma \vdash \lambda x : T_1. t : T_1 \rightarrow T_2} \\
(T_{\text{pair}}) \frac{\Delta; \Gamma \vdash t_1 : T_1 \quad \Delta; \Gamma \vdash t_2 : T_2}{\Delta; \Gamma \vdash \langle t_1, t_2 \rangle : T_1 \times T_2} \quad (T_\Lambda) \frac{\Delta, X; \Gamma \vdash t : T \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \Lambda X. t : \forall X. T} \\
(T_{\text{asc}}) \frac{\Delta; \Gamma \vdash t : T' \quad T' = T}{\Delta; \Gamma \vdash t :: T : T} \quad (T_{\text{app}}) \frac{\Delta; \Gamma \vdash t_1 : T_1 \quad \Delta; \Gamma \vdash t_2 : T_2 \quad \text{dom}(T_1) = T_2}{\Delta; \Gamma \vdash t_1 t_2 : \text{cod}(T_1)} \\
(T_{\text{pair}i}) \frac{\Delta; \Gamma \vdash t : T}{\Delta; \Gamma \vdash \pi_i(T) : \text{proj}_i(T)} \quad (T_{\text{app}G}) \frac{\Delta; \Gamma \vdash t : T \quad \Delta \vdash F}{\Delta; \Gamma \vdash t [F] : \text{inst}(T, F)}
\end{array}$$

$$\begin{array}{cccc}
\text{dom} : \text{TYPE} \rightarrow \text{TYPE} & \text{cod} : \text{TYPE} \rightarrow \text{TYPE} & \text{proj}_i : \text{TYPE} \rightarrow \text{TYPE} & \text{inst} : \text{TYPE}^2 \rightarrow \text{TYPE} \\
\text{dom}(T_1 \rightarrow T_2) = T_1 & \text{cod}(T_1 \rightarrow T_2) = T_2 & \text{proj}_i(T_1 \times T_2) = T_i & \text{inst}(\forall X. T, T') = T[T'/X] \setminus X \\
\text{dom}(T) \text{ undefined o/w} & \text{cod}(T) \text{ undefined o/w} & \text{proj}_i(T) \text{ undefined o/w} & \text{inst}(T, T') \text{ undefined o/w}
\end{array}$$

$$\begin{array}{c}
B[T'/X] \setminus X = B \\
X[T'/X] \setminus X = T' \\
Y[T'/X] \setminus X = Y \\
(T_1 \rightarrow T_2)[T'/X] \setminus X = T_1[T'/X] \setminus X \rightarrow T_2[T'/X] \setminus X \\
(T_1 \times T_2)[T'/X] \setminus X = T_1[T'/X] \setminus X \times T_2[T'/X] \setminus X \\
(\forall Y. T)[T'/X] \setminus X = \forall Y. T[T'/X] \setminus X
\end{array}$$

Figure B.3: System F_1 : Syntax and Static Semantics

Lemma B.92 (α is Sound) *If A is not empty, then $A \subseteq \gamma(\alpha(A))$.*

PROOF. By induction on the structure of the non-empty set A .

Case ($\{\mathbf{B}\}$). Then $\alpha(A) = \alpha(\{\mathbf{B}\}) = \mathbf{B}$ and $\gamma(\mathbf{B}) = \{\mathbf{B}\}$. Therefore, $A = \{\mathbf{B}\} \subseteq \{\mathbf{B}\} = \gamma(\alpha(A))$, as we are required to prove.

Case ($\{\mathbf{X}\}$). Then $\alpha(A) = \alpha(\{\mathbf{X}\}) = \mathbf{X}$ and $\gamma(\mathbf{X}) = \{\mathbf{X}\}$. Therefore, $A = \{\mathbf{X}\} \subseteq \{\mathbf{X}\} = \gamma(\alpha(A))$, as we are required to prove.

Case ($\{\overline{T_{i1} \rightarrow T_{i2}}\}$). Then $\alpha(\{\overline{T_{i1} \rightarrow T_{i2}}\}) = \alpha(\{\overline{T_{i1}}\}) \rightarrow \alpha(\{\overline{T_{i2}}\})$. We know that

$$\gamma(\alpha(\{\overline{T_{i1}}\}) \rightarrow \alpha(\{\overline{T_{i2}}\})) = \gamma(\alpha(\{\overline{T_{i1}}\})) \rightarrow \gamma(\alpha(\{\overline{T_{i2}}\})) = \{T_1 \rightarrow T_2 \mid T_1 \in \gamma(\alpha(\{\overline{T_{i1}}\})), T_2 \in \gamma(\alpha(\{\overline{T_{i2}}\}))\}$$

By the inductive hypotheses, on $\{\overline{T_{i1}}\}$ and $\{\overline{T_{i2}}\}$, we get that $\{\overline{T_{i1}}\} \subseteq \gamma(\alpha(\{\overline{T_{i1}}\}))$ and $\{\overline{T_{i2}}\} \subseteq \gamma(\alpha(\{\overline{T_{i2}}\}))$. Therefore, we get that $\{\overline{T_{i1} \rightarrow T_{i2}}\} \subseteq \{\overline{T_{i1}}\} \rightarrow \{\overline{T_{i2}}\} \subseteq \gamma(\alpha(\{\overline{T_{i1}}\})) \rightarrow \gamma(\alpha(\{\overline{T_{i2}}\}))$, as we are required to prove.

Case ($\{\overline{T_{i1} \times T_{i2}}\}$). Then $\alpha(\{\overline{T_{i1} \times T_{i2}}\}) = \alpha(\{\overline{T_{i1}}\}) \times \alpha(\{\overline{T_{i2}}\})$. We know that

$$\gamma(\alpha(\{\overline{T_{i1}}\}) \times \alpha(\{\overline{T_{i2}}\})) = \gamma(\alpha(\{\overline{T_{i1}}\})) \times \gamma(\alpha(\{\overline{T_{i2}}\})) = \{T_1 \times T_2 \mid T_1 \in \gamma(\alpha(\{\overline{T_{i1}}\})), T_2 \in \gamma(\alpha(\{\overline{T_{i2}}\}))\}$$

$t \longrightarrow t$ **Notion of reduction**

$$\begin{aligned}
v &:: T \longrightarrow v \\
(\lambda x : T. t) v &\longrightarrow t[v/x] \\
\pi_i(\langle v_1, v_2 \rangle) &\longrightarrow v_i \\
(\Lambda X. t)[F] &\longrightarrow t[F/X]
\end{aligned}$$

$t \mapsto t$ or **error** **Evaluation frames and reduction**

$$\begin{aligned}
f &::= \square :: T \mid \square t \mid v \square \mid \langle \square, t \rangle \mid \langle v, \square \rangle \mid \pi_i(\square) \mid \square [F] \\
(R \rightarrow) \frac{t \longrightarrow t'}{t \mapsto t'} & \qquad (Rf) \frac{t \mapsto t'}{f[t] \mapsto f[t']} \\
(Rerr) \frac{t \longrightarrow \mathbf{error}}{t \mapsto \mathbf{error}} & \qquad (Rferr) \frac{t \mapsto \mathbf{error}}{f[t] \mapsto \mathbf{error}}
\end{aligned}$$

Figure B.4: System F_1 : Dynamic Semantics

$$\begin{array}{ll}
\gamma(\mathbf{B}) = \{ B \} & \alpha(\{ B \}) = \mathbf{B} \\
\gamma(\mathbf{X}) = \{ X \} & \alpha(\{ X \}) = \mathbf{X} \\
\gamma(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}_2) & \alpha(\{\overline{T_1} \rightarrow \overline{T_2}\}) = \alpha(\{\overline{T_1}\}) \rightarrow \alpha(\{\overline{T_2}\}) \\
\gamma(\mathbf{G}_1 \times \mathbf{G}_2) = \gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}_2) & \alpha(\{\overline{T_1} \times \overline{T_2}\}) = \alpha(\{\overline{T_1}\}) \times \alpha(\{\overline{T_2}\}) \\
\gamma(\forall X. \mathbf{G}_1) = \forall X. \gamma(\mathbf{G}_1) & \alpha(\{\overline{\forall X. T_1}\}) = \forall X. \alpha(\{\overline{T_1}\}) \\
\gamma(?_\delta) = \{ T \mid \mathbf{ftv}(T) \subseteq \mathbf{ftv}(\delta) \} & \alpha(\{\overline{T}\}) = ?_\delta \qquad \text{where } \mathbf{ftv}(\delta) = \mathbf{ftv}(\{\overline{T}\})
\end{array}$$

Figure B.5: Type concretization (γ) and abstraction (α)

By the inductive hypotheses, on $\{\overline{T_{i1}}\}$ and $\{\overline{T_{i2}}\}$, we get that $\{\overline{T_{i1}}\} \subseteq \gamma(\alpha(\{\overline{T_{i1}}\}))$ and $\{\overline{T_{i2}}\} \subseteq \gamma(\alpha(\{\overline{T_{i2}}\}))$. Therefore, we get that $\{\overline{T_{i1} \times T_{i2}}\} \subseteq \{\overline{T_{i1}}\} \times \{\overline{T_{i2}}\} \subseteq \gamma(\alpha(\{\overline{T_{i1}}\})) \times \gamma(\alpha(\{\overline{T_{i2}}\}))$, as we are required to prove.

Case ($\{\overline{\forall X. T_{i1}}\}$). Then $\alpha(\{\overline{\forall X. T_{i1}}\}) = \forall X. \alpha(\{\overline{T_{i1}}\})$. We know that

$$\gamma(\forall X. \alpha(\{\overline{T_{i1}}\})) = \forall X. \gamma(\alpha(\{\overline{T_{i1}}\})) = \{\forall X. T_1 \mid T_1 \in \gamma(\alpha(\{\overline{T_{i1}}\}))\}$$

By the inductive hypotheses, on $\{\overline{T_{i1}}\}$, we get that $\{\overline{T_{i1}}\} \subseteq \gamma(\alpha(\{\overline{T_{i1}}\}))$. Therefore, we get that $\{\overline{\forall X. T_{i1}}\} = \forall X. \{\overline{T_{i1}}\} \subseteq \forall X. \gamma(\alpha(\{\overline{T_{i1}}\}))$, as we are required to prove.

Case ($\{\overline{T_i}\}$ heterogeneous). Then $\alpha(\{\overline{T_i}\}) = ?_\delta$, where $\mathbf{ftv}(\delta) = \mathbf{ftv}(\{\overline{T_i}\})$. Then, we know that $\gamma(\alpha(A)) = \gamma(?_\delta) = \{T \mid \mathbf{ftv}(T) \subseteq \mathbf{ftv}(\delta)\}$. Since $\mathbf{ftv}(\{\overline{T_i}\}) = \mathbf{ftv}(\delta) = \mathbf{ftv}(?_\delta)$, we get that $A \subseteq \gamma(?_\delta)$.

□

Lemma B.93 (α is Optimal) *If A is not empty and $A \subseteq \gamma(\mathbf{G})$ then $\alpha(A) \sqsubseteq \mathbf{G}$.*

PROOF. We proceed by induction on \mathbf{G} .

Case (B). We know that $\gamma(\mathbf{B}) = \{\mathbf{B}\}$. Since A is no empty and $A \subseteq \gamma(\mathbf{G})$, then we know that $A = \gamma(\mathbf{B}) = \{\mathbf{B}\}$. Also, we know that $\alpha(\{\mathbf{B}\}) = \mathbf{B} \sqsubseteq \mathbf{B} = \mathbf{G}$, as we are required to prove.

Case (X). We know that $\gamma(\mathbf{X}) = \{\mathbf{X}\}$. Since A is no empty and $A \subseteq \gamma(\mathbf{G})$, then we know that $A = \gamma(\mathbf{X}) = \{\mathbf{X}\}$. Also, we know that $\alpha(\{\mathbf{X}\}) = \mathbf{X} \sqsubseteq \mathbf{X} = \mathbf{G}$, as we are required to prove.

Case ($\mathbf{G}_1 \rightarrow \mathbf{G}_2$). We know that $\gamma(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}_2)$. Since A is no empty and $A \subseteq \gamma(\mathbf{G})$, then we know that $A = \{\overline{T_{11} \rightarrow T_{12}}\}$, $\{\overline{T_{11}}\} \subseteq \gamma(\mathbf{G}_1)$ and $\{\overline{T_{12}}\} \subseteq \gamma(\mathbf{G}_2)$. We know that $\alpha(A) = \alpha(\{\overline{T_{11} \rightarrow T_{12}}\}) = \alpha(\{\overline{T_{11}}\}) \rightarrow \alpha(\{\overline{T_{12}}\})$. We are required to prove that $\alpha(\{\overline{T_{11}}\}) \rightarrow \alpha(\{\overline{T_{12}}\}) \sqsubseteq \mathbf{G}_1 \rightarrow \mathbf{G}_2$. Therefore, by $(\sqsubseteq_{\rightarrow})$, we are required to show that $\alpha(\{\overline{T_{11}}\}) \sqsubseteq \mathbf{G}_1$ and $\alpha(\{\overline{T_{12}}\}) \sqsubseteq \mathbf{G}_2$, which follows immediately by the inductive hypothesis on \mathbf{G}_1 and \mathbf{G}_2 , with $\{\overline{T_{11}}\} \subseteq \gamma(\mathbf{G}_1)$ and $\{\overline{T_{12}}\} \subseteq \gamma(\mathbf{G}_2)$, noting that $\{\overline{T_{11}}\}$ and $\{\overline{T_{12}}\}$ are not empty set because $\{\overline{T_{11} \rightarrow T_{12}}\}$ is not empty.

Case ($\mathbf{G}_1 \times \mathbf{G}_2$). We know that $\gamma(\mathbf{G}_1 \times \mathbf{G}_2) = \gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}_2)$. Since A is no empty and $A \subseteq \gamma(\mathbf{G})$, then we know that $A = \{\overline{T_{11} \times T_{12}}\}$, $\{\overline{T_{11}}\} \subseteq \gamma(\mathbf{G}_1)$ and $\{\overline{T_{12}}\} \subseteq \gamma(\mathbf{G}_2)$. We know that $\alpha(A) = \alpha(\{\overline{T_{11} \times T_{12}}\}) = \alpha(\{\overline{T_{11}}\}) \times \alpha(\{\overline{T_{12}}\})$. We are required to prove that $\alpha(\{\overline{T_{11}}\}) \times \alpha(\{\overline{T_{12}}\}) \sqsubseteq \mathbf{G}_1 \times \mathbf{G}_2$. Therefore, by (\sqsubseteq_{\times}) , we are required to show that $\alpha(\{\overline{T_{11}}\}) \sqsubseteq \mathbf{G}_1$ and $\alpha(\{\overline{T_{12}}\}) \sqsubseteq \mathbf{G}_2$, which follows immediately by the inductive hypothesis on \mathbf{G}_1 and \mathbf{G}_2 , with $\{\overline{T_{11}}\} \subseteq \gamma(\mathbf{G}_1)$ and $\{\overline{T_{12}}\} \subseteq \gamma(\mathbf{G}_2)$, noting that $\{\overline{T_{11}}\}$ and $\{\overline{T_{12}}\}$ are not empty set because $\{\overline{T_{11} \times T_{12}}\}$ is not empty.

Case ($\forall \mathbf{X}. \mathbf{G}_1$). We know that $\gamma(\forall \mathbf{X}. \mathbf{G}_1) = \forall \mathbf{X}. \gamma(\mathbf{G}_1)$. Since A is no empty and $A \subseteq \gamma(\mathbf{G})$, then we know that $A = \{\overline{\forall \mathbf{X}. T_{11}}\}$, $\{\overline{T_{11}}\} \subseteq \gamma(\mathbf{G}_1)$. We know that $\alpha(A) = \alpha(\{\overline{\forall \mathbf{X}. T_{11}}\}) = \forall \mathbf{X}. \alpha(\{\overline{T_{11}}\})$. We are required to prove that $\forall \mathbf{X}. \alpha(\{\overline{T_{11}}\}) \sqsubseteq \forall \mathbf{X}. \mathbf{G}_1$. Therefore, by (\sqsubseteq_{\forall}) , we are required to show that $\alpha(\{\overline{T_{11}}\}) \sqsubseteq \mathbf{G}_1$, which follows immediately by the inductive hypothesis on \mathbf{G}_1 , with $\{\overline{T_{11}}\} \subseteq \gamma(\mathbf{G}_1)$, noting that $\{\overline{T_{11}}\}$ is not empty set because $\{\overline{\forall \mathbf{X}. T_{11}}\}$ is not empty.

Case ($?_{\delta}$). We know that $\gamma(?_{\delta}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. Since $A \subseteq \gamma(\mathbf{G})$, by Lemma B.98, we know that $\text{ftv}(A) \subseteq \text{ftv}(?_{\delta}) = \text{ftv}(\delta)$. By Lemma B.101, we get that $\text{ftv}(A) = \text{ftv}(\alpha(A))$ and thus, $\text{ftv}(\alpha(A)) \subseteq \text{ftv}(?_{\delta})$. Therefore, by Lemma B.100, we get that $\alpha(A) \sqsubseteq ?_{\delta}$, as we are required to show.

□

$$\alpha(\gamma(\mathbf{G})) = \mathbf{G}.$$

PROOF. We know that $\gamma(\mathbf{G}) \sqsubseteq \gamma(\mathbf{G})$. Thus, by Lemma B.93, we get that $\alpha(\gamma(\mathbf{G})) \sqsubseteq \mathbf{G}$. By Lemma B.92, we get that $\gamma(\mathbf{G}) \subseteq \gamma(\alpha(\gamma(\mathbf{G})))$, thus, by Definition B.94, we know that $\mathbf{G} \sqsubseteq \alpha(\gamma(\mathbf{G}))$. Since $\alpha(\gamma(\mathbf{G})) \sqsubseteq \mathbf{G}$ and $\mathbf{G} \sqsubseteq \alpha(\gamma(\mathbf{G}))$, we know that $\alpha(\gamma(\mathbf{G})) = \mathbf{G}$. □

Definition B.94 $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$ if and only if $\gamma(\mathbf{G}_1) \subseteq \gamma(\mathbf{G}_2)$.

[Precision, inductively] The inductive definition of type precision given in Figure B.6 is equivalent to Definition B.94.

PROOF. We first prove that $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2 \implies \gamma(\mathbf{G}_1) \subseteq \gamma(\mathbf{G}_2)$. We proceed by induction on \mathbf{G}_1 .

Case ($\mathbf{G}_1 = \mathbf{B}$). Since $\mathbf{B} \sqsubseteq \mathbf{G}_2$, we know that $\mathbf{G}_2 = \mathbf{B}$ or $\mathbf{G}_2 = ?_{\delta}$.

- If $\mathbf{G}_2 = \mathbf{B}$, then we have to prove that $\gamma(\mathbf{B}) = \{\mathbf{B}\} \subseteq \{\mathbf{B}\} = \gamma(\mathbf{B})$, which follows immediately.
- If $\mathbf{G}_2 = ?_{\delta}$, then we have to prove that $\gamma(\mathbf{B}) = \{\mathbf{B}\} \subseteq \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\} = \gamma(?_{\delta})$, which follows immediately.

Case ($\mathbf{G}_1 = \mathbf{X}$). Since $\mathbf{X} \sqsubseteq \mathbf{G}_2$, we know that $\mathbf{G}_2 = \mathbf{X}$ or $\mathbf{G}_2 = ?_\delta$.

- If $\mathbf{G}_2 = \mathbf{X}$, then we have to prove that $\gamma(\mathbf{X}) = \{\mathbf{X}\} \subseteq \{\mathbf{X}\} = \gamma(\mathbf{X})$, which follows immediately.
- If $\mathbf{G}_2 = ?_\delta$, by ($\sqsubseteq_{\mathbf{X}?$), we know that $\mathbf{X} : \mathbf{X} \in \delta$, or what is the same $\mathbf{X} \in \text{ftv}(\delta)$. We have to prove that $\gamma(\mathbf{X}) = \{\mathbf{X}\} \subseteq \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\} = \gamma(?_\delta)$, which follows immediately since $\mathbf{X} \in \text{ftv}(\delta)$.

Case ($\mathbf{G}_1 = \mathbf{G}_{11} \rightarrow \mathbf{G}_{12}$). We know that $\gamma(\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}) = \gamma(\mathbf{G}_{11}) \rightarrow \gamma(\mathbf{G}_{12}) = \{T_{11} \rightarrow T_{12} \mid T_{11} \in \gamma(\mathbf{G}_{11}) \wedge T_{12} \in \gamma(\mathbf{G}_{12})\}$. Since $(\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}) \sqsubseteq \mathbf{G}_2$, we know that $\mathbf{G}_2 = \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}$ or $\mathbf{G}_2 = ?_\delta$.

- We have that $\mathbf{G}_2 = \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}$, such that $\mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$, by ($\sqsubseteq_{\rightarrow}$) rule. Therefore, by the induction hypotheses on \mathbf{G}_{11} and \mathbf{G}_{12} , we know that $\gamma(\mathbf{G}_{11}) \subseteq \gamma(\mathbf{G}_{21})$ and $\gamma(\mathbf{G}_{12}) \subseteq \gamma(\mathbf{G}_{22})$, where $\gamma(\mathbf{G}_{11}) = \{T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \gamma(\mathbf{G}_{21}) = \{T_{21} \in \gamma(\mathbf{G}_{21})\}$ and $\gamma(\mathbf{G}_{12}) = \{T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \gamma(\mathbf{G}_{22}) = \{T_{22} \in \gamma(\mathbf{G}_{22})\}$. We know that $\gamma(\mathbf{G}_{21} \rightarrow \mathbf{G}_{22}) = \gamma(\mathbf{G}_{21}) \rightarrow \gamma(\mathbf{G}_{22}) = \{T_{21} \rightarrow T_{22} \mid T_{21} \in \gamma(\mathbf{G}_{21}) \wedge T_{22} \in \gamma(\mathbf{G}_{22})\}$. We are required to prove that $\gamma(\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}) \subseteq \gamma(\mathbf{G}_{21} \rightarrow \mathbf{G}_{22})$, or what is the same $\{T_{11} \rightarrow T_{12} \mid T_{11} \in \gamma(\mathbf{G}_{11}) \wedge T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \{T_{21} \rightarrow T_{22} \mid T_{21} \in \gamma(\mathbf{G}_{21}) \wedge T_{22} \in \gamma(\mathbf{G}_{22})\}$. Thus, we are required to prove that $\gamma(\mathbf{G}_{11}) \subseteq \gamma(\mathbf{G}_{21})$ and $\gamma(\mathbf{G}_{12}) \subseteq \gamma(\mathbf{G}_{22})$, which we already know.
- We have that $\mathbf{G}_2 = ?_\delta$, and we know that $\mathbf{G}_{11} \sqsubseteq ?_\delta$ and $\mathbf{G}_{12} \sqsubseteq ?_\delta$, by ($\sqsubseteq_{\rightarrow?}$) rule. Therefore, by the induction hypotheses on \mathbf{G}_{11} and \mathbf{G}_{12} , we know that $\gamma(\mathbf{G}_{11}) \subseteq \gamma(?_\delta)$ and $\gamma(\mathbf{G}_{12}) \subseteq \gamma(?_\delta)$, where $\gamma(\mathbf{G}_{11}) = \{T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \gamma(?_\delta) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$ and $\gamma(\mathbf{G}_{12}) = \{T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \gamma(?_\delta) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. We are required to prove that $\gamma(\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}) \subseteq \gamma(?_\delta)$, or what is the same $\{T_{11} \rightarrow T_{12} \mid T_{11} \in \gamma(\mathbf{G}_{11}) \wedge T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. Thus, we are required to prove that $\gamma(\mathbf{G}_{11}) \subseteq \gamma(?_\delta)$ and $\gamma(\mathbf{G}_{12}) \subseteq \gamma(?_\delta)$, which we already know.

Case ($\mathbf{G}_1 = \mathbf{G}_{11} \times \mathbf{G}_{12}$). We know that $\gamma(\mathbf{G}_{11} \times \mathbf{G}_{12}) = \gamma(\mathbf{G}_{11}) \times \gamma(\mathbf{G}_{12}) = \{T_{11} \times T_{12} \mid T_{11} \in \gamma(\mathbf{G}_{11}) \wedge T_{12} \in \gamma(\mathbf{G}_{12})\}$. Since $(\mathbf{G}_{11} \times \mathbf{G}_{12}) \sqsubseteq \mathbf{G}_2$, we know that $\mathbf{G}_2 = \mathbf{G}_{21} \times \mathbf{G}_{22}$ or $\mathbf{G}_2 = ?_\delta$.

- We have that $\mathbf{G}_2 = \mathbf{G}_{21} \times \mathbf{G}_{22}$, such that $\mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$, by (\sqsubseteq_{\times}) rule. Therefore, by the induction hypotheses on \mathbf{G}_{11} and \mathbf{G}_{12} , we know that $\gamma(\mathbf{G}_{11}) \subseteq \gamma(\mathbf{G}_{21})$ and $\gamma(\mathbf{G}_{12}) \subseteq \gamma(\mathbf{G}_{22})$, where $\gamma(\mathbf{G}_{11}) = \{T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \gamma(\mathbf{G}_{21}) = \{T_{21} \in \gamma(\mathbf{G}_{21})\}$ and $\gamma(\mathbf{G}_{12}) = \{T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \gamma(\mathbf{G}_{22}) = \{T_{22} \in \gamma(\mathbf{G}_{22})\}$. We know that $\gamma(\mathbf{G}_{21} \times \mathbf{G}_{22}) = \gamma(\mathbf{G}_{21}) \times \gamma(\mathbf{G}_{22}) = \{T_{21} \times T_{22} \mid T_{21} \in \gamma(\mathbf{G}_{21}) \wedge T_{22} \in \gamma(\mathbf{G}_{22})\}$. We are required to prove that $\gamma(\mathbf{G}_{11} \times \mathbf{G}_{12}) \subseteq \gamma(\mathbf{G}_{21} \times \mathbf{G}_{22})$, or what is the same $\{T_{11} \times T_{12} \mid T_{11} \in \gamma(\mathbf{G}_{11}) \wedge T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \{T_{21} \times T_{22} \mid T_{21} \in \gamma(\mathbf{G}_{21}) \wedge T_{22} \in \gamma(\mathbf{G}_{22})\}$. Thus, we are required to prove that $\gamma(\mathbf{G}_{11}) \subseteq \gamma(\mathbf{G}_{21})$ and $\gamma(\mathbf{G}_{12}) \subseteq \gamma(\mathbf{G}_{22})$, which we already know.
- We have that $\mathbf{G}_2 = ?_\delta$ and we know that $\mathbf{G}_{11} \sqsubseteq ?_\delta$ and $\mathbf{G}_{12} \sqsubseteq ?_\delta$, by ($\sqsubseteq_{\times?}$) rule. Therefore, by the induction hypotheses on \mathbf{G}_{11} and \mathbf{G}_{12} , we know that $\gamma(\mathbf{G}_{11}) \subseteq \gamma(?_\delta)$ and $\gamma(\mathbf{G}_{12}) \subseteq \gamma(?_\delta)$, where $\gamma(\mathbf{G}_{11}) = \{T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \gamma(?_\delta) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$ and $\gamma(\mathbf{G}_{12}) = \{T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \gamma(?_\delta) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. We are required to prove that $\gamma(\mathbf{G}_{11} \times \mathbf{G}_{12}) \subseteq \gamma(?_\delta)$, or what is the same $\{T_{11} \times T_{12} \mid T_{11} \in \gamma(\mathbf{G}_{11}) \wedge T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. Thus, we are required to prove that $\gamma(\mathbf{G}_{11}) \subseteq \gamma(?_\delta)$ and $\gamma(\mathbf{G}_{12}) \subseteq \gamma(?_\delta)$, which we already know.

Case ($\mathbf{G}_1 = \forall \mathbf{X}. \mathbf{G}_{11}$). We know that $\gamma(\forall \mathbf{X}. \mathbf{G}_{11}) = \forall \mathbf{X}. \gamma(\mathbf{G}_{11}) = \{\forall \mathbf{X}. T_{11} \mid T_{11} \in \gamma(\mathbf{G}_{11})\}$. Since $(\forall \mathbf{X}. \mathbf{G}_{11}) \sqsubseteq \mathbf{G}_2$, we know that $\mathbf{G}_2 = \forall \mathbf{X}. \mathbf{G}_{21}$ or $\mathbf{G}_2 = ?_\delta$.

- We have that $\mathbf{G}_2 = \forall \mathbf{X}.\mathbf{G}_{21}$, such that $\mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$, by (\sqsubseteq_{\forall}) rule. Therefore, by the induction hypotheses on \mathbf{G}_{11} , we know that $\gamma(\mathbf{G}_{11}) \subseteq \gamma(\mathbf{G}_{21})$, where $\gamma(\mathbf{G}_{11}) = \{T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \gamma(\mathbf{G}_{21}) = \{T_{21} \in \gamma(\mathbf{G}_{21})\}$. We know that $\gamma(\forall \mathbf{X}.\mathbf{G}_{21}) = \forall \mathbf{X}.\gamma(\mathbf{G}_{21}) = \{\forall \mathbf{X}.T_{21} \mid T_{21} \in \gamma(\mathbf{G}_{21})\}$. We are required to prove that $\gamma(\forall \mathbf{X}.\mathbf{G}_{11}) \subseteq \gamma(\forall \mathbf{X}.\mathbf{G}_{21})$, or what is the same $\{\forall \mathbf{X}.T_{11} \mid T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \{\forall \mathbf{X}.T_{21} \mid T_{21} \in \gamma(\mathbf{G}_{21})\}$. Thus, we are required to prove that $\gamma(\mathbf{G}_{11}) \subseteq \gamma(\mathbf{G}_{21})$ which we already know.
- We have that $\mathbf{G}_2 = ?_{\delta}$, and we know that $\mathbf{G}_{11} \sqsubseteq ?_{\delta, \mathbf{X}:\mathbf{X}}$, by $(\sqsubseteq_{\forall?})$ rule. Therefore, by the induction hypotheses on \mathbf{G}_{11} , we know that $\gamma(\mathbf{G}_{11}) \subseteq \gamma(?_{\delta, \mathbf{X}:\mathbf{X}})$, where $\gamma(\mathbf{G}_{11}) = \{T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \gamma(?_{\delta, \mathbf{X}:\mathbf{X}}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta, \mathbf{X} : \mathbf{X})\}$ and we are required to prove that $\gamma(\forall \mathbf{X}.\mathbf{G}_{11}) \subseteq \gamma(?_{\delta})$, or what is the same $\{\forall \mathbf{X}.T_{11} \mid T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. Thus, we are required to prove that $\gamma(\mathbf{G}_{11}) \subseteq \gamma(?_{\delta, \mathbf{X}:\mathbf{X}})$, which we already know.

Case ($\mathbf{G}_1 = ?_{\delta}$). Since $?_{\delta} \sqsubseteq \mathbf{G}_2$, we know that $\mathbf{G}_2 = ?_{\delta'}$ and $\delta \subseteq \delta'$, by $(\sqsubseteq_{?})$. Therefore, we know that $\text{ftv}(\delta) \subseteq \text{ftv}(\delta')$. We are required to show that $\gamma(?_{\delta}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\} \subseteq \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta')\} = \gamma(\mathbf{G}_2)$, which follows by $\text{ftv}(\delta) \subseteq \text{ftv}(\delta')$.

Next, we prove that $\gamma(\mathbf{G}_1) \subseteq \gamma(\mathbf{G}_2) \implies \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$. We proceed by induction on \mathbf{G}_1 .

Case ($\mathbf{G}_1 = \mathbf{B}$). Since $\gamma(\mathbf{B}) \subseteq \gamma(\mathbf{G}_2)$, we know that $\mathbf{G}_2 = \mathbf{B}$ or $\mathbf{G}_2 = ?_{\delta}$.

- If $\mathbf{G}_2 = \mathbf{B}$, then we have to prove that $\mathbf{B} \sqsubseteq \mathbf{B}$, which is trivial by $(\sqsubseteq_{\mathbf{B}})$.
- If $\mathbf{G}_2 = ?_{\delta}$, then we have to prove that $\mathbf{B} \sqsubseteq ?_{\delta}$, which is trivial by $(\sqsubseteq_{\mathbf{B} ?})$ rule.

Case ($\mathbf{G}_1 = \mathbf{X}$). Since $\gamma(\mathbf{X}) \subseteq \gamma(\mathbf{G}_2)$, we know that $\mathbf{G}_2 = \mathbf{X}$ or $\mathbf{G}_2 = ?_{\delta}$.

- If $\mathbf{G}_2 = \mathbf{X}$, then we have to prove that $\mathbf{X} \sqsubseteq \mathbf{X}$, which is trivial by $(\sqsubseteq_{\mathbf{X}})$.
- If $\mathbf{G}_2 = ?_{\delta}$, since $\gamma(\mathbf{X}) = \{\mathbf{X}\} \subseteq \gamma(\mathbf{G}_2) = \gamma(?_{\delta}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$, we know that $\mathbf{X} \in \text{ftv}(\delta)$, and therefore $\mathbf{X} : \mathbf{X} \in \delta$. We have to prove that $\mathbf{X} \sqsubseteq ?_{\delta}$, which is trivial by $(\sqsubseteq_{\mathbf{X} ?})$ rule and $\mathbf{X} : \mathbf{X} \in \delta$.

Case ($\mathbf{G}_1 = \mathbf{G}_{11} \rightarrow \mathbf{G}_{12}$). We know that $\gamma(\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}) = \gamma(\mathbf{G}_{11}) \rightarrow \gamma(\mathbf{G}_{12}) = \{T_{11} \rightarrow T_{12} \mid T_{11} \in \gamma(\mathbf{G}_{11}) \wedge T_{12} \in \gamma(\mathbf{G}_{12})\}$. Since $\gamma(\mathbf{G}_{11} \rightarrow \mathbf{G}_{12}) \subseteq \gamma(\mathbf{G}_2)$, we know that $\mathbf{G}_2 = \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}$ or $\mathbf{G}_2 = ?_{\delta}$.

- We have that $\mathbf{G}_2 = \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}$, such that $\gamma(\mathbf{G}_{21} \rightarrow \mathbf{G}_{22}) = \gamma(\mathbf{G}_{21}) \rightarrow \gamma(\mathbf{G}_{22}) = \{T_{21} \rightarrow T_{22} \mid T_{21} \in \gamma(\mathbf{G}_{21}) \wedge T_{22} \in \gamma(\mathbf{G}_{22})\}$ and $\{T_{11} \rightarrow T_{12} \mid T_{11} \in \gamma(\mathbf{G}_{11}) \wedge T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \{T_{21} \rightarrow T_{22} \mid T_{21} \in \gamma(\mathbf{G}_{21}) \wedge T_{22} \in \gamma(\mathbf{G}_{22})\}$. Thus, we know that $\gamma(\mathbf{G}_{11}) = \{T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \gamma(\mathbf{G}_{21}) = \{T_{21} \in \gamma(\mathbf{G}_{21})\}$ and $\gamma(\mathbf{G}_{12}) = \{T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \gamma(\mathbf{G}_{22}) = \{T_{22} \in \gamma(\mathbf{G}_{22})\}$. Therefore, by the induction hypotheses on \mathbf{G}_{11} and \mathbf{G}_{12} , we know that $\mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$. Therefore, by $(\sqsubseteq_{\rightarrow})$ rule, $\mathbf{G}_{11} \rightarrow \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}$, and the result holds.
- We have that $\mathbf{G}_2 = ?_{\delta}$, such that $\gamma(?_{\delta}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. Thus, we know that $\gamma(\mathbf{G}_{11}) = \{T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \gamma(?_{\delta}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$ and $\gamma(\mathbf{G}_{12}) = \{T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \gamma(?_{\delta}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. Therefore, by the induction hypotheses on \mathbf{G}_{11} and \mathbf{G}_{12} , we know that $\mathbf{G}_{11} \sqsubseteq ?_{\delta}$ and $\mathbf{G}_{12} \sqsubseteq ?_{\delta}$. By $(\sqsubseteq_{\rightarrow ?})$ rule, we are required to prove that $\mathbf{G}_{11} \rightarrow \mathbf{G}_{12} \sqsubseteq ?_{\delta} \rightarrow ?_{\delta}$, or what is the same, we are required to show that $\mathbf{G}_{11} \sqsubseteq ?_{\delta}$ and $\mathbf{G}_{12} \sqsubseteq ?_{\delta}$, which we already know.

Case ($\mathbf{G}_1 = \mathbf{G}_{11} \times \mathbf{G}_{12}$). We know that $\gamma(\mathbf{G}_{11} \times \mathbf{G}_{12}) = \gamma(\mathbf{G}_{11}) \times \gamma(\mathbf{G}_{12}) = \{T_{11} \times T_{12} \mid T_{11} \in \gamma(\mathbf{G}_{11}) \wedge T_{12} \in \gamma(\mathbf{G}_{12})\}$. Since $\gamma(\mathbf{G}_{11} \times \mathbf{G}_{12}) \subseteq \gamma(\mathbf{G}_2)$, we know that $\mathbf{G}_2 = \mathbf{G}_{21} \times \mathbf{G}_{22}$ or $\mathbf{G}_2 = ?_{\delta}$.

- We have that $\mathbf{G}_2 = \mathbf{G}_{21} \times \mathbf{G}_{22}$, such that $\gamma(\mathbf{G}_{21} \times \mathbf{G}_{22}) = \gamma(\mathbf{G}_{21}) \times \gamma(\mathbf{G}_{22}) = \{T_{21} \times T_{22} \mid T_{21} \in \gamma(\mathbf{G}_{21}) \wedge T_{22} \in \gamma(\mathbf{G}_{22})\}$ and $\{T_{11} \times T_{12} \mid T_{11} \in \gamma(\mathbf{G}_{11}) \wedge T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \{T_{21} \times T_{22} \mid T_{21} \in \gamma(\mathbf{G}_{21}) \wedge T_{22} \in \gamma(\mathbf{G}_{22})\}$. Thus, we know that $\gamma(\mathbf{G}_{11}) = \{T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \gamma(\mathbf{G}_{21}) = \{T_{21} \in \gamma(\mathbf{G}_{21})\}$ and $\gamma(\mathbf{G}_{12}) = \{T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \gamma(\mathbf{G}_{22}) = \{T_{22} \in \gamma(\mathbf{G}_{22})\}$. Therefore, by the induction hypotheses on \mathbf{G}_{11} and \mathbf{G}_{12} , we know that $\mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$. Therefore, by (\sqsubseteq_{\times}) rule, $\mathbf{G}_{11} \times \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{21} \times \mathbf{G}_{22}$, and the result holds.
- We have that $\mathbf{G}_2 = ?_{\delta}$, such that $\gamma(?_{\delta}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. Thus, we know that $\gamma(\mathbf{G}_{11}) = \{T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \gamma(?_{\delta}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$ and $\gamma(\mathbf{G}_{12}) = \{T_{12} \in \gamma(\mathbf{G}_{12})\} \subseteq \gamma(?_{\delta}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. Therefore, by the induction hypotheses on \mathbf{G}_{11} and \mathbf{G}_{12} , we know that $\mathbf{G}_{11} \sqsubseteq ?_{\delta}$ and $\mathbf{G}_{12} \sqsubseteq ?_{\delta}$. By $(\sqsubseteq_{\times?})$ rule, we are required to prove that $\mathbf{G}_{11} \times \mathbf{G}_{12} \sqsubseteq ?_{\delta} \times ?_{\delta}$, or what is the same, we are required to show that $\mathbf{G}_{11} \sqsubseteq ?_{\delta}$ and $\mathbf{G}_{12} \sqsubseteq ?_{\delta}$, which we already know.

Case ($\mathbf{G}_1 = \forall \mathbf{X}.\mathbf{G}_{11}$). We know that $\gamma(\forall \mathbf{X}.\mathbf{G}_{11}) = \forall \mathbf{X}.\gamma(\mathbf{G}_{11}) = \{\forall \mathbf{X}.T_{11} \mid T_{11} \in \gamma(\mathbf{G}_{11})\}$. Since $\gamma(\forall \mathbf{X}.\mathbf{G}_{11}) \subseteq \gamma(\mathbf{G}_2)$, we know that $\mathbf{G}_2 = \forall \mathbf{X}.\mathbf{G}_{21}$ or $\mathbf{G}_2 = ?_{\delta}$.

- We have that $\mathbf{G}_2 = \forall \mathbf{X}.\mathbf{G}_{21}$, such that $\gamma(\forall \mathbf{X}.\mathbf{G}_{21}) = \forall \mathbf{X}.\gamma(\mathbf{G}_{21}) = \{\forall \mathbf{X}.T_{21} \mid T_{21} \in \gamma(\mathbf{G}_{21})\}$ and $\{\forall \mathbf{X}.T_{11} \mid T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \{\forall \mathbf{X}.T_{21} \mid T_{21} \in \gamma(\mathbf{G}_{21})\}$. Thus, we know that $\gamma(\mathbf{G}_{11}) = \{T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \gamma(\mathbf{G}_{21}) = \{T_{21} \in \gamma(\mathbf{G}_{21})\}$. Therefore, by the induction hypotheses on \mathbf{G}_{11} , we know that $\mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$. Therefore, by (\sqsubseteq_{\forall}) rule, $\forall \mathbf{X}.\mathbf{G}_{11} \sqsubseteq \forall \mathbf{X}.\mathbf{G}_{21}$, and the result holds.
- We have that $\mathbf{G}_2 = ?_{\delta}$, such that $\gamma(?_{\delta}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. Thus, we know that $\gamma(\mathbf{G}_{11}) = \{T_{11} \in \gamma(\mathbf{G}_{11})\} \subseteq \gamma(?_{\delta, \mathbf{X}:\mathbf{X}}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta, \mathbf{X}:\mathbf{X})\}$. Therefore, by the induction hypotheses on \mathbf{G}_{11} , we know that $\mathbf{G}_{11} \sqsubseteq ?_{\delta, \mathbf{X}:\mathbf{X}}$. By $(\sqsubseteq_{\forall?})$ rule, we are required to prove that $\forall \mathbf{X}.\mathbf{G}_{11} \sqsubseteq \forall \mathbf{X}?._{\delta, \mathbf{X}:\mathbf{X}}$, or what is the same, we are required to show that $\mathbf{G}_{11} \sqsubseteq ?_{\delta, \mathbf{X}:\mathbf{X}}$, which we already know.

Case ($\mathbf{G}_1 = ?_{\delta}$). Since $\gamma(?_{\delta}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\} \subseteq \gamma(\mathbf{G}_2)$, we know that $\mathbf{G}_2 = ?_{\delta'}$, where $\gamma(?_{\delta'}) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta')\}$. Therefore, we have that $\text{ftv}(\delta) \subseteq \text{ftv}(\delta')$, or what is the same $\delta \subseteq \delta'$. We are required to prove that $?_{\delta} \sqsubseteq ?_{\delta'}$, which follows by $(\sqsubseteq_?)$ rule and $\delta \subseteq \delta'$.

□

Lemma B.95

1. $\mathbf{B} \sqcap \mathbf{B} = \mathbf{B}$;
2. $\mathbf{X} \sqcap \mathbf{X} = \mathbf{X}$;
3. $(\mathbf{G}_1 \rightarrow \mathbf{G}'_1) \sqcap (\mathbf{G}_2 \rightarrow \mathbf{G}'_2) = (\mathbf{G}_1 \sqcap \mathbf{G}_2) \rightarrow (\mathbf{G}'_1 \sqcap \mathbf{G}'_2)$;
4. $(\mathbf{G}_1 \times \mathbf{G}'_1) \sqcap (\mathbf{G}_2 \times \mathbf{G}'_2) = (\mathbf{G}_1 \sqcap \mathbf{G}_2) \times (\mathbf{G}'_1 \sqcap \mathbf{G}'_2)$;
5. $(\forall \mathbf{X}.\mathbf{G}_1) \sqcap (\forall \mathbf{X}.\mathbf{G}_2) = \forall \mathbf{X}.\mathbf{G}_1 \sqcap \mathbf{G}_2$;
6. $?_{\delta_1} \sqcap ?_{\delta_2} = ?_{\delta_1 \cap \delta_2}$;
7. $\mathbf{B} \sqcap ?_{\delta} = ?_{\delta} \sqcap \mathbf{B} = \mathbf{B}$;
8. $\mathbf{X} \sqcap ?_{\delta} = ?_{\delta} \sqcap \mathbf{X} = \mathbf{X}$ if $\mathbf{X} : \mathbf{X} \in \delta$;
9. $(\mathbf{G}_1 \rightarrow \mathbf{G}'_1) \sqcap ?_{\delta} = ?_{\delta} \sqcap (\mathbf{G}_1 \rightarrow \mathbf{G}'_1) = (\mathbf{G}_1 \sqcap ?_{\delta}) \rightarrow (\mathbf{G}'_1 \sqcap ?_{\delta})$;

$$10. (\mathbf{G}_1 \times \mathbf{G}'_1) \sqcap ?_\delta = ?_\delta \sqcap (\mathbf{G}_1 \times \mathbf{G}'_1) = (\mathbf{G}_1 \sqcap ?_\delta) \times (\mathbf{G}'_1 \sqcap ?_\delta);$$

$$11. (\forall \mathbf{X}. \mathbf{G}_1) \sqcap ?_\delta = ?_\delta \sqcap (\forall \mathbf{X}. \mathbf{G}_1) = \forall \mathbf{X}. (\mathbf{G}_1 \sqcap ?_{\delta, \mathbf{X}: \mathbf{X}});$$

12. $\mathbf{G}_1 \sqcap \mathbf{G}_2$ is undefined otherwise.

PROOF.

1. $\mathbf{B} \sqcap \mathbf{B} = \mathbf{B}$. Since $\gamma(\mathbf{B}) = \{\mathbf{B}\}$ and $\alpha(\{\mathbf{B}\}) = \mathbf{B}$, we get that $\mathbf{B} \sqcap \mathbf{B} = \alpha(\gamma(\mathbf{B}) \cap \gamma(\mathbf{B})) = \alpha(\{\mathbf{B}\} \cap \{\mathbf{B}\}) = \alpha(\{\mathbf{B}\}) = \mathbf{B}$, as we are required to show.

2. $\mathbf{X} \sqcap \mathbf{X} = \mathbf{X}$. Since $\gamma(\mathbf{X}) = \{\mathbf{X}\}$ and $\alpha(\{\mathbf{X}\}) = \mathbf{X}$, we get that $\mathbf{X} \sqcap \mathbf{X} = \alpha(\gamma(\mathbf{X}) \cap \gamma(\mathbf{X})) = \alpha(\{\mathbf{X}\} \cap \{\mathbf{X}\}) = \alpha(\{\mathbf{X}\}) = \mathbf{X}$, as we are required to show.

3. $(\mathbf{G}_1 \rightarrow \mathbf{G}'_1) \sqcap (\mathbf{G}_2 \rightarrow \mathbf{G}'_2) = (\mathbf{G}_1 \sqcap \mathbf{G}_2) \rightarrow (\mathbf{G}'_1 \sqcap \mathbf{G}'_2)$. We know that

$$\begin{aligned} (\mathbf{G}_1 \rightarrow \mathbf{G}'_1) \sqcap (\mathbf{G}_2 \rightarrow \mathbf{G}'_2) &= \alpha(\gamma(\mathbf{G}_1 \rightarrow \mathbf{G}'_1) \cap \gamma(\mathbf{G}_2 \rightarrow \mathbf{G}'_2)) = \alpha((\gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}'_1)) \cap (\gamma(\mathbf{G}_2) \rightarrow \gamma(\mathbf{G}'_2))) = \\ &= \alpha((\gamma(\mathbf{G}_1) \cap \gamma(\mathbf{G}_2)) \rightarrow (\gamma(\mathbf{G}'_1) \cap \gamma(\mathbf{G}'_2))) = \alpha(\gamma(\mathbf{G}_1) \cap \gamma(\mathbf{G}_2)) \rightarrow \alpha(\gamma(\mathbf{G}'_1) \cap \gamma(\mathbf{G}'_2)) = (\mathbf{G}_1 \sqcap \mathbf{G}_2) \rightarrow (\mathbf{G}'_1 \sqcap \mathbf{G}'_2) \end{aligned}$$

as we are required to show. Note that $(A \cap B) \times (C \cap D) = (A \times C) \cap (B \times D)$, by set theory.

4. $(\mathbf{G}_1 \times \mathbf{G}'_1) \sqcap (\mathbf{G}_2 \times \mathbf{G}'_2) = (\mathbf{G}_1 \sqcap \mathbf{G}_2) \times (\mathbf{G}'_1 \sqcap \mathbf{G}'_2)$. We know that

$$\begin{aligned} (\mathbf{G}_1 \times \mathbf{G}'_1) \sqcap (\mathbf{G}_2 \times \mathbf{G}'_2) &= \alpha(\gamma(\mathbf{G}_1 \times \mathbf{G}'_1) \cap \gamma(\mathbf{G}_2 \times \mathbf{G}'_2)) = \alpha((\gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}'_1)) \cap (\gamma(\mathbf{G}_2) \times \gamma(\mathbf{G}'_2))) = \\ &= \alpha((\gamma(\mathbf{G}_1) \cap \gamma(\mathbf{G}_2)) \times (\gamma(\mathbf{G}'_1) \cap \gamma(\mathbf{G}'_2))) = \alpha(\gamma(\mathbf{G}_1) \cap \gamma(\mathbf{G}_2)) \times \alpha(\gamma(\mathbf{G}'_1) \cap \gamma(\mathbf{G}'_2)) = (\mathbf{G}_1 \sqcap \mathbf{G}_2) \times (\mathbf{G}'_1 \sqcap \mathbf{G}'_2) \end{aligned}$$

as we are required to show. Note that $(A \cap B) \times (C \cap D) = (A \times C) \cap (B \times D)$, by set theory.

5. $(\forall \mathbf{X}. \mathbf{G}_1) \sqcap (\forall \mathbf{X}. \mathbf{G}_2) = \forall \mathbf{X}. (\mathbf{G}_1 \sqcap \mathbf{G}_2)$. We know that

$$\begin{aligned} (\forall \mathbf{X}. \mathbf{G}_1) \sqcap (\forall \mathbf{X}. \mathbf{G}_2) &= \alpha(\gamma(\forall \mathbf{X}. \mathbf{G}_1) \cap \gamma(\forall \mathbf{X}. \mathbf{G}_2)) = \alpha((\forall \mathbf{X}. \gamma(\mathbf{G}_1)) \cap (\forall \mathbf{X}. \gamma(\mathbf{G}_2))) = \\ &= \alpha(\forall \mathbf{X}. (\gamma(\mathbf{G}_1) \cap \gamma(\mathbf{G}_2))) = \forall \mathbf{X}. \alpha(\gamma(\mathbf{G}_1) \cap \gamma(\mathbf{G}_2)) = \forall \mathbf{X}. (\mathbf{G}_1 \sqcap \mathbf{G}_2) \end{aligned}$$

as we are required to show.

6. $?_{\delta_1} \sqcap ?_{\delta_2} = ?_{\delta_1 \cap \delta_2}$. We know that

$$\begin{aligned} ?_{\delta_1} \sqcap ?_{\delta_2} &= \alpha(\gamma(?_{\delta_1}) \cap \gamma(?_{\delta_2})) = \alpha(\{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta_1)\} \cap \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta_2)\}) = \\ &= \alpha(\{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta_1) \cap \text{ftv}(\delta_2)\}) = \alpha(\{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta_1 \cap \delta_2)\}) = ?_{\delta_1 \cap \delta_2} \end{aligned}$$

7. $\mathbf{B} \sqcap ?_\delta = ?_\delta \sqcap \mathbf{B} = \mathbf{B}$. We know that $\gamma(\mathbf{B}) = \{\mathbf{B}\}$, $\alpha(\{\mathbf{B}\}) = \mathbf{B}$, $\{\mathbf{B}\} \subseteq \gamma(?_\delta)$ and $\{\mathbf{B}\} \cap \gamma(?_\delta) = \{\mathbf{B}\}$. Thus, $\mathbf{B} \sqcap ?_\delta = \alpha(\gamma(\mathbf{B}) \cap \gamma(?_\delta)) = \alpha(\{\mathbf{B}\} \cap \gamma(?_\delta)) = \alpha(\{\mathbf{B}\}) = \mathbf{B}$ and $?_\delta \sqcap \mathbf{B} = \alpha(\gamma(?_\delta) \cap \gamma(\mathbf{B})) = \alpha(\gamma(?_\delta) \cap \{\mathbf{B}\}) = \alpha(\{\mathbf{B}\}) = \mathbf{B}$, as we are required to show.

8. $\mathbf{X} \sqcap ?_\delta = ?_\delta \sqcap \mathbf{X} = \mathbf{X}$ if $\mathbf{X} : \mathbf{X} \in \delta$. We know that $\gamma(\mathbf{X}) = \{\mathbf{X}\}$, $\alpha(\{\mathbf{X}\}) = \mathbf{X}$, $\{\mathbf{X}\} \subseteq \gamma(?_\delta)$ ($\mathbf{X} : \mathbf{X} \in \delta$ and $\text{ftv}(\mathbf{X}) = \{\mathbf{X}\} \subseteq \text{ftv}(\delta)$) and $\{\mathbf{X}\} \cap \gamma(?_\delta) = \{\mathbf{X}\}$. Thus, $\mathbf{X} \sqcap ?_\delta = \alpha(\gamma(\mathbf{X}) \cap \gamma(?_\delta)) = \alpha(\{\mathbf{X}\} \cap \gamma(?_\delta)) = \alpha(\{\mathbf{X}\}) = \mathbf{X}$ and $?_\delta \sqcap \mathbf{X} = \alpha(\gamma(?_\delta) \cap \gamma(\mathbf{X})) = \alpha(\gamma(?_\delta) \cap \{\mathbf{X}\}) = \alpha(\{\mathbf{X}\}) = \mathbf{X}$, as we are required to show.

9. $(\mathbf{G}_1 \rightarrow \mathbf{G}'_1) \sqcap ?_\delta = ?_\delta \sqcap (\mathbf{G}_1 \rightarrow \mathbf{G}'_1) = (\mathbf{G}_1 \sqcap ?_\delta) \rightarrow (\mathbf{G}'_1 \sqcap ?_\delta)$. We know that $(\gamma(?_\delta) \rightarrow \gamma(?_\delta)) \subseteq \gamma(?_\delta)$ and $(\gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}'_1)) \sqcap \gamma(?_\delta) = (\gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}'_1)) \sqcap (\gamma(?_\delta) \rightarrow \gamma(?_\delta))$. Thus, we get that

$$\begin{aligned} (\mathbf{G}_1 \rightarrow \mathbf{G}'_1) \sqcap ?_\delta &= \alpha(\gamma(\mathbf{G}_1 \rightarrow \mathbf{G}'_1) \sqcap \gamma(?_\delta)) = \alpha((\gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}'_1)) \sqcap (\gamma(?_\delta) \rightarrow \gamma(?_\delta))) = \\ &\alpha((\gamma(\mathbf{G}_1) \sqcap \gamma(?_\delta)) \rightarrow (\gamma(\mathbf{G}'_1) \sqcap \gamma(?_\delta))) = \alpha((\gamma(\mathbf{G}_1) \sqcap \gamma(?_\delta)) \rightarrow \alpha((\gamma(\mathbf{G}'_1) \sqcap \gamma(?_\delta)))) = (\mathbf{G}_1 \sqcap ?_\delta) \rightarrow (\mathbf{G}'_1 \sqcap ?_\delta) \end{aligned}$$

and

$$\begin{aligned} ?_\delta \sqcap (\mathbf{G}_1 \rightarrow \mathbf{G}'_1) &= \\ \alpha(\gamma(?_\delta) \sqcap \gamma(\mathbf{G}_1 \rightarrow \mathbf{G}'_1)) &= \alpha((\gamma(?_\delta) \rightarrow \gamma(?_\delta)) \sqcap (\gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}'_1))) = \alpha((\gamma(?_\delta) \sqcap \gamma(\mathbf{G}_1)) \rightarrow (\gamma(?_\delta) \sqcap \gamma(\mathbf{G}'_1))) = \\ \alpha((\gamma(?_\delta) \sqcap \gamma(\mathbf{G}_1)) \rightarrow \alpha((\gamma(?_\delta) \sqcap \gamma(\mathbf{G}'_1)))) &= \alpha((\gamma(\mathbf{G}_1) \sqcap \gamma(?_\delta)) \rightarrow \alpha((\gamma(\mathbf{G}'_1) \sqcap \gamma(?_\delta)))) = (\mathbf{G}_1 \sqcap ?_\delta) \rightarrow (\mathbf{G}'_1 \sqcap ?_\delta) \end{aligned}$$

10. $(\mathbf{G}_1 \times \mathbf{G}'_1) \sqcap ?_\delta = ?_\delta \sqcap (\mathbf{G}_1 \times \mathbf{G}'_1) = (\mathbf{G}_1 \sqcap ?_\delta) \times (\mathbf{G}'_1 \sqcap ?_\delta)$. We know that $(\gamma(?_\delta) \times \gamma(?_\delta)) \subseteq \gamma(?_\delta)$ and $(\gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}'_1)) \sqcap \gamma(?_\delta) = (\gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}'_1)) \sqcap (\gamma(?_\delta) \times \gamma(?_\delta))$. Thus, we get that

$$\begin{aligned} (\mathbf{G}_1 \times \mathbf{G}'_1) \sqcap ?_\delta &= \alpha(\gamma(\mathbf{G}_1 \times \mathbf{G}'_1) \sqcap \gamma(?_\delta)) = \alpha((\gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}'_1)) \sqcap (\gamma(?_\delta) \times \gamma(?_\delta))) = \\ \alpha((\gamma(\mathbf{G}_1) \sqcap \gamma(?_\delta)) \times (\gamma(\mathbf{G}'_1) \sqcap \gamma(?_\delta))) &= \alpha((\gamma(\mathbf{G}_1) \sqcap \gamma(?_\delta)) \times \alpha((\gamma(\mathbf{G}'_1) \sqcap \gamma(?_\delta)))) = (\mathbf{G}_1 \sqcap ?_\delta) \times (\mathbf{G}'_1 \sqcap ?_\delta) \end{aligned}$$

and

$$\begin{aligned} ?_\delta \sqcap (\mathbf{G}_1 \times \mathbf{G}'_1) &= \\ \alpha(\gamma(?_\delta) \sqcap \gamma(\mathbf{G}_1 \times \mathbf{G}'_1)) &= \alpha((\gamma(?_\delta) \times \gamma(?_\delta)) \sqcap (\gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}'_1))) = \alpha((\gamma(?_\delta) \sqcap \gamma(\mathbf{G}_1)) \times (\gamma(?_\delta) \sqcap \gamma(\mathbf{G}'_1))) = \\ \alpha((\gamma(?_\delta) \sqcap \gamma(\mathbf{G}_1)) \times \alpha((\gamma(?_\delta) \sqcap \gamma(\mathbf{G}'_1)))) &= \alpha((\gamma(\mathbf{G}_1) \sqcap \gamma(?_\delta)) \times \alpha((\gamma(\mathbf{G}'_1) \sqcap \gamma(?_\delta)))) = (\mathbf{G}_1 \sqcap ?_\delta) \times (\mathbf{G}'_1 \sqcap ?_\delta) \end{aligned}$$

11. $(\forall \mathbf{X}. \mathbf{G}_1) \sqcap ?_\delta = ?_\delta \sqcap (\forall \mathbf{X}. \mathbf{G}_1) = \forall \mathbf{X}. (\mathbf{G}_1 \sqcap ?_\delta, \mathbf{X} : \mathbf{X})$. We know that $(\forall \mathbf{X}. \gamma(?_\delta, \mathbf{X} : \mathbf{X})) \subseteq \gamma(?_\delta)$ and $(\forall \mathbf{X}. \gamma(\mathbf{G}_1)) \sqcap \gamma(?_\delta) = (\forall \mathbf{X}. \gamma(\mathbf{G}_1)) \sqcap (\forall \mathbf{X}. \gamma(?_\delta, \mathbf{X} : \mathbf{X}))$. Thus, we get that

$$\begin{aligned} (\forall \mathbf{X}. \mathbf{G}_1) \sqcap ?_\delta &= \alpha(\gamma(\forall \mathbf{X}. \mathbf{G}_1) \sqcap \gamma(?_\delta)) = \alpha((\forall \mathbf{X}. \gamma(\mathbf{G}_1)) \sqcap (\forall \mathbf{X}. \gamma(?_\delta, \mathbf{X} : \mathbf{X}))) = \\ \alpha(\forall \mathbf{X}. (\gamma(\mathbf{G}_1) \sqcap \gamma(?_\delta, \mathbf{X} : \mathbf{X}))) &= \forall \mathbf{X}. \alpha((\gamma(\mathbf{G}_1) \sqcap \gamma(?_\delta, \mathbf{X} : \mathbf{X}))) = \forall \mathbf{X}. (\mathbf{G}_1 \sqcap ?_\delta, \mathbf{X} : \mathbf{X}) \end{aligned}$$

and

$$\begin{aligned} ?_\delta \sqcap (\forall \mathbf{X}. \mathbf{G}_1) &= \alpha(\gamma(?_\delta) \sqcap \gamma(\forall \mathbf{X}. \mathbf{G}_1)) = \alpha((\forall \mathbf{X}. \gamma(?_\delta, \mathbf{X} : \mathbf{X})) \sqcap (\forall \mathbf{X}. \gamma(\mathbf{G}_1))) = \alpha(\forall \mathbf{X}. (\gamma(?_\delta, \mathbf{X} : \mathbf{X}) \sqcap \gamma(\mathbf{G}_1))) = \\ \forall \mathbf{X}. \alpha((\gamma(?_\delta, \mathbf{X} : \mathbf{X}) \sqcap \gamma(\mathbf{G}_1))) &= \forall \mathbf{X}. \alpha((\gamma(\mathbf{G}_1) \sqcap \gamma(?_\delta, \mathbf{X} : \mathbf{X}))) = \forall \mathbf{X}. (\mathbf{G}_1 \sqcap ?_\delta, \mathbf{X} : \mathbf{X}) \end{aligned}$$

12. $\mathbf{G}_1 \sqcap \mathbf{G}_2$ is undefined otherwise. If we try to calculate $\mathbf{G}_1 \sqcap \mathbf{G}_2 = \alpha(\gamma(\mathbf{G}_1) \sqcap \gamma(\mathbf{G}_2))$, we will get that $\gamma(\mathbf{G}_1) \sqcap \gamma(\mathbf{G}_2) = \emptyset$, therefore, $\alpha(\gamma(\mathbf{G}_1) \sqcap \gamma(\mathbf{G}_2))$ is not defined and the result holds.

□

Definition B.96 (Consistency) $\mathbf{G}_1 \sim \mathbf{G}_2$ if and only if there exists T_1 and T_2 such that $T_1 = T_2$, $T_1 \in \gamma(\mathbf{G}_1)$ and $T_2 \in \gamma(\mathbf{G}_2)$.

[Consistency, inductively] The inductive definition of type consistency given in Figure B.6 is equivalent to Definition B.96.

PROOF. We first prove that if $\mathbf{G} \sim \mathbf{G}'$ by definition in Figure B.6, we get that $\mathbf{G} \sim \mathbf{G}'$, by Definition B.96. We proceed by induction on $\mathbf{G} \sim \mathbf{G}'$.

Case ($\mathbf{B} \sim \mathbf{B}$). We are required to prove that $\mathbf{B} \sim \mathbf{B}$, or what is the same that there exists T_1 and T_2 such that $T_1 = T_2$, $T_1 \in \gamma(\mathbf{G}) = \gamma(\mathbf{B}) = \{\mathbf{B}\}$ and $T_2 \in \gamma(\mathbf{G}') = \gamma(\mathbf{B}) = \{\mathbf{B}\}$. Thus, if we choose $T_1 = T_2 = \mathbf{B}$, the result holds.

Case ($\mathbf{X} \sim \mathbf{X}$). We are required to prove that $\mathbf{X} \sim \mathbf{X}$, or what is the same that there exists T_1 and T_2 such that $T_1 = T_2$, $T_1 \in \gamma(\mathbf{G}) = \gamma(\mathbf{X}) = \{\mathbf{X}\}$ and $T_2 \in \gamma(\mathbf{G}') = \gamma(\mathbf{X}) = \{\mathbf{X}\}$. Thus, if we choose $T_1 = T_2 = \mathbf{X}$, the result holds.

Case ($\mathbf{G}_1 \rightarrow \mathbf{G}_2 \sim \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$). By the inductive hypothesis on $\mathbf{G}_1 \sim \mathbf{G}'_1$ and $\mathbf{G}_2 \sim \mathbf{G}'_2$, we get that that there exists T_1 and T'_1 , and T_2 and T'_2 such that $T_1 = T'_1$, $T_1 \in \gamma(\mathbf{G}_1)$, $T'_1 \in \gamma(\mathbf{G}'_1)$, $T_2 = T'_2$, $T_2 \in \gamma(\mathbf{G}_2)$ and $T'_2 \in \gamma(\mathbf{G}'_2)$. We are required to prove that that there exists $T_1 \rightarrow T_2$ and $T'_1 \rightarrow T'_2$ such that $T_1 \rightarrow T_2 = T'_1 \rightarrow T'_2$, $T_1 \rightarrow T_2 \in \gamma(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}_2)$ and $T'_1 \rightarrow T'_2 \in \gamma(\mathbf{G}'_1 \rightarrow \mathbf{G}'_2) = \gamma(\mathbf{G}'_1) \rightarrow \gamma(\mathbf{G}'_2)$, as we have already showed.

Case ($\mathbf{G}_1 \times \mathbf{G}_2 \sim \mathbf{G}'_1 \times \mathbf{G}'_2$). By the inductive hypothesis on $\mathbf{G}_1 \sim \mathbf{G}'_1$ and $\mathbf{G}_2 \sim \mathbf{G}'_2$, we get that that there exists T_1 and T'_1 , and T_2 and T'_2 such that $T_1 = T'_1$, $T_1 \in \gamma(\mathbf{G}_1)$, $T'_1 \in \gamma(\mathbf{G}'_1)$, $T_2 = T'_2$, $T_2 \in \gamma(\mathbf{G}_2)$ and $T'_2 \in \gamma(\mathbf{G}'_2)$. We are required to prove that that there exists $T_1 \times T_2$ and $T'_1 \times T'_2$ such that $T_1 \times T_2 = T'_1 \times T'_2$, $T_1 \times T_2 \in \gamma(\mathbf{G}_1 \times \mathbf{G}_2) = \gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}_2)$ and $T'_1 \times T'_2 \in \gamma(\mathbf{G}'_1 \times \mathbf{G}'_2) = \gamma(\mathbf{G}'_1) \times \gamma(\mathbf{G}'_2)$, as we have already showed.

Case ($\forall \mathbf{X}. \mathbf{G}_1 \sim \forall \mathbf{X}. \mathbf{G}'_1$). By the inductive hypothesis on $\mathbf{G}_1 \sim \mathbf{G}'_1$, we get that that there exists T_1 and T'_1 , $T_1 = T'_1$, $T_1 \in \gamma(\mathbf{G}_1)$ and $T'_1 \in \gamma(\mathbf{G}'_1)$. We are required to prove that that there exists $\forall \mathbf{X}. T_1$ and $\forall \mathbf{X}. T'_1$ such that $\forall \mathbf{X}. T_1 = \forall \mathbf{X}. T'_1$, $\forall \mathbf{X}. T_1 \in \gamma(\forall \mathbf{X}. \mathbf{G}_1) = \forall \mathbf{X}. \gamma(\mathbf{G}_1)$ and $\forall \mathbf{X}. T'_1 \in \gamma(\forall \mathbf{X}. \mathbf{G}'_1) = \forall \mathbf{X}. \gamma(\mathbf{G}'_1)$, as we have already showed.

Case ($\mathbf{B} \sim ?_\delta$). We are required to prove that $\mathbf{B} \sim ?_\delta$, or what is the same that there exists T_1 and T_2 such that $T_1 = T_2$, $T_1 \in \gamma(\mathbf{G}) = \gamma(\mathbf{B}) = \{\mathbf{B}\}$ and $T_2 \in \gamma(\mathbf{G}') = \gamma(?_\delta) \supseteq \{\mathbf{B}\}$. Thus, if we choose $T_1 = T_2 = \mathbf{B}$, the result holds.

Case ($?_\delta \sim \mathbf{B}$). Identically, to the inverse case.

Case ($\mathbf{X} \sim ?_\delta \wedge \mathbf{X} : \mathbf{X} \in \delta$). We are required to prove that $\mathbf{X} \sim ?_\delta$, or what is the same that there exists T_1 and T_2 such that $T_1 = T_2$, $T_1 \in \gamma(\mathbf{G}) = \gamma(\mathbf{X}) = \{\mathbf{X}\}$ and $T_2 \in \gamma(\mathbf{G}') = \gamma(?_\delta) \supseteq \{\mathbf{X}\}$ (since $\mathbf{X} : \mathbf{X} \in \delta$ and $\text{ftv}(\mathbf{X}) = \{\mathbf{X}\} \subseteq \text{ftv}(\delta)$). Thus, if we choose $T_1 = T_2 = \mathbf{X}$, the result holds.

Case ($?_\delta \sim \mathbf{X} \wedge \mathbf{X} : \mathbf{X} \in \delta$). Identically, to the inverse case.

Case ($\mathbf{G}_1 \rightarrow \mathbf{G}_2 \sim ?_\delta$). By the inductive hypothesis on $\mathbf{G}_1 \sim ?_\delta$ and $\mathbf{G}_2 \sim ?_\delta$, we get that that there exists T_1 and T'_1 , and T_2 and T'_2 such that $T_1 = T'_1$, $T_1 \in \gamma(\mathbf{G}_1)$, $T'_1 \in \gamma(?_\delta)$, $T_2 = T'_2$, $T_2 \in \gamma(\mathbf{G}_2)$ and $T'_2 \in \gamma(?_\delta)$. We are required to prove that that there exists $T_1 \rightarrow T_2$ and $T'_1 \rightarrow T'_2$ such that $T_1 \rightarrow T_2 = T'_1 \rightarrow T'_2$, $T_1 \rightarrow T_2 \in \gamma(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}_2)$ and $T'_1 \rightarrow T'_2 \in \gamma(?_\delta) \supseteq \gamma(?_\delta) \rightarrow \gamma(?_\delta)$, as we have already showed.

Case ($\mathbf{G}_1 \times \mathbf{G}_2 \sim ?_\delta$). By the inductive hypothesis on $\mathbf{G}_1 \sim ?_\delta$ and $\mathbf{G}_2 \sim ?_\delta$, we get that that there exists T_1 and T'_1 , and T_2 and T'_2 such that $T_1 = T'_1$, $T_1 \in \gamma(\mathbf{G}_1)$, $T'_1 \in \gamma(?_\delta)$, $T_2 = T'_2$, $T_2 \in \gamma(\mathbf{G}_2)$ and $T'_2 \in \gamma(?_\delta)$. We are required to prove that that there exists $T_1 \times T_2$ and $T'_1 \times T'_2$ such that $T_1 \times T_2 = T'_1 \times T'_2$, $T_1 \times T_2 \in \gamma(\mathbf{G}_1 \times \mathbf{G}_2) = \gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}_2)$ and $T'_1 \times T'_2 \in \gamma(?_\delta) \supseteq \gamma(?_\delta) \times \gamma(?_\delta)$, as we have already showed.

Case ($\forall \mathbf{X}. \mathbf{G}_1 \sim ?_\delta$). By the inductive hypothesis on $\mathbf{G}_1 \sim ?_\delta, \mathbf{X} : \mathbf{X}$, we get that that there exists T_1 and T'_1 such that $T_1 = T'_1$, $T_1 \in \gamma(\mathbf{G}_1)$ and $T'_1 \in \gamma(?_\delta, \mathbf{X} : \mathbf{X})$. We are required to prove that that there exists $\forall \mathbf{X}. T_1$ and $\forall \mathbf{X}. T'_1$ such that $\forall \mathbf{X}. T_1 = \forall \mathbf{X}. T'_1$, $\forall \mathbf{X}. T_1 \in \gamma(\forall \mathbf{X}. \mathbf{G}_1) = \forall \mathbf{X}. \gamma(\mathbf{G}_1)$ and $\forall \mathbf{X}. T'_1 \in \gamma(?_\delta) \supseteq \forall \mathbf{X}. \gamma(?_\delta, \mathbf{X} : \mathbf{X})$, as we have already showed.

Case ($?_\delta \sim \mathbf{G}_1 \rightarrow \mathbf{G}_2$). Identically, to the inverse case.

Case ($?_\delta \sim \mathbf{G}_1 \times \mathbf{G}_2$). Identically, to the inverse case.

Case ($?_\delta \sim \forall \mathbf{X}. \mathbf{G}$). Identically, to the inverse case.

Case ($?_\delta \sim ?_{\delta'}$). We know that $\mathbf{B} \in \gamma(?_\delta)$ and $\mathbf{B} \in \gamma(?_{\delta'})$, thus, the result holds.

Next, we prove that if $\mathbf{G} \sim \mathbf{G}'$, by Definition B.96, we get that $\mathbf{G} \sim \mathbf{G}'$, by definition in Figure B.6. By Definition B.96, we know that $\mathbf{G} \sim \mathbf{G}'$ if there exists T and T' such that $T = T' T \in \gamma(\mathbf{G})$ and $T' \in \gamma(\mathbf{G}')$. We proceed by induction on T .

Case (B). Then, we know that $\mathbf{B} \in \gamma(\mathbf{G})$ and $\mathbf{B} \in \gamma(\mathbf{G}')$. Thus, we have the following cases.

- $\mathbf{G} = \mathbf{B}$ and $\mathbf{G}' = \mathbf{B}$. The result holds immediately, since $\mathbf{B} \sim \mathbf{B}$.
- $\mathbf{G} = \mathbf{B}$ and $\mathbf{G}' = ?_\delta$. The result holds immediately, since $\mathbf{B} \sim ?_\delta$.
- $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \mathbf{B}$. The result holds immediately, since $?_\delta \sim \mathbf{B}$.
- $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = ?_{\delta'}$. The result holds immediately, since $?_\delta \sim ?_{\delta'}$.

Case (X). Then, we know that $\mathbf{X} \in \gamma(\mathbf{G})$ and $\mathbf{X} \in \gamma(\mathbf{G}')$. Thus, we have the following cases.

- $\mathbf{G} = \mathbf{X}$ and $\mathbf{G}' = \mathbf{X}$. The result holds immediately, since $\mathbf{X} \sim \mathbf{X}$.
- $\mathbf{G} = \mathbf{X}$, $\mathbf{G}' = ?_\delta$ and $\mathbf{X} : \mathbf{X} \in \delta$. The result holds immediately, since $\mathbf{X} \sim ?_\delta$.
- $\mathbf{G} = ?_\delta$, $\mathbf{G}' = \mathbf{X}$ and $\mathbf{X} : \mathbf{X} \in \delta$. The result holds immediately, since $?_\delta \sim \mathbf{X}$.
- $\mathbf{G} = ?_\delta$, $\mathbf{G}' = ?_{\delta'}$, $\mathbf{X} : \mathbf{X} \in \delta$ and $\mathbf{X} : \mathbf{X} \in \delta'$. The result holds immediately, since $?_\delta \sim ?_{\delta'}$.

Case ($T_1 \rightarrow T_2$). Then, we know that $T_1 \rightarrow T_2 \in \gamma(\mathbf{G})$ and $T_1 \rightarrow T_2 \in \gamma(\mathbf{G}')$. Thus, we have the following cases.

- $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\mathbf{G}' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$. Thus, we know that $T_1 \rightarrow T_2 \in \gamma(\mathbf{G}_1 \rightarrow \mathbf{G}_2)$ and $T_1 \rightarrow T_2 \in \gamma(\mathbf{G}'_1 \rightarrow \mathbf{G}'_2)$. Therefore, we get that $T_1 \in \gamma(\mathbf{G}_1)$, $T_2 \in \gamma(\mathbf{G}_2)$, $T_1 \in \gamma(\mathbf{G}'_1)$ and $T_2 \in \gamma(\mathbf{G}'_2)$. By the inductive hypothesis, we get that $\mathbf{G}_1 \sim \mathbf{G}'_1$ and $\mathbf{G}_2 \sim \mathbf{G}'_2$. Therefore, the result holds immediately, since $\mathbf{G}_1 \rightarrow \mathbf{G}_2 \sim \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$.
- $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$ and $\mathbf{G}' = ?_\delta$. Thus, we know that $T_1 \rightarrow T_2 \in \gamma(\mathbf{G}_1 \rightarrow \mathbf{G}_2)$ and $T_1 \rightarrow T_2 \in \gamma(?_\delta) \supseteq \gamma(?_\delta) \rightarrow \gamma(?_\delta)$. Therefore, we get that $T_1 \in \gamma(\mathbf{G}_1)$, $T_2 \in \gamma(\mathbf{G}_2)$, $T_1 \in \gamma(?_\delta)$ and $T_2 \in \gamma(?_\delta)$. By the inductive hypothesis, we get that $\mathbf{G}_1 \sim ?_\delta$ and $\mathbf{G}_2 \sim ?_\delta$. Therefore, the result holds immediately, since $\mathbf{G}_1 \rightarrow \mathbf{G}_2 \sim ?_\delta$.
- $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$. Identically to the previous case.
- $\mathbf{G} = ?_\delta$ and $\mathbf{G}' = ?_{\delta'}$. The result holds immediately, since $?_\delta \sim ?_{\delta'}$.

Case ($T_1 \times T_2$). Then, we know that $T_1 \times T_2 \in \gamma(\mathbf{G})$ and $T_1 \times T_2 \in \gamma(\mathbf{G}')$. Thus, we have the following cases.

- $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$ and $\mathbf{G}' = \mathbf{G}'_1 \times \mathbf{G}'_2$. Thus, we know that $T_1 \times T_2 \in \gamma(\mathbf{G}_1 \times \mathbf{G}_2)$ and $T_1 \times T_2 \in \gamma(\mathbf{G}'_1 \times \mathbf{G}'_2)$. Therefore, we get that $T_1 \in \gamma(\mathbf{G}_1)$, $T_2 \in \gamma(\mathbf{G}_2)$, $T_1 \in \gamma(\mathbf{G}'_1)$ and $T_2 \in \gamma(\mathbf{G}'_2)$. By the inductive hypothesis, we get that $\mathbf{G}_1 \sim \mathbf{G}'_1$ and $\mathbf{G}_2 \sim \mathbf{G}'_2$. Therefore, the result holds immediately, since $\mathbf{G}_1 \times \mathbf{G}_2 \sim \mathbf{G}'_1 \times \mathbf{G}'_2$.
- $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$ and $\mathbf{G}' = ?_\delta$. Thus, we know that $T_1 \times T_2 \in \gamma(\mathbf{G}_1 \times \mathbf{G}_2)$ and $T_1 \times T_2 \in \gamma(?_\delta) \supseteq \gamma(?_\delta) \times \gamma(?_\delta)$. Therefore, we get that $T_1 \in \gamma(\mathbf{G}_1)$, $T_2 \in \gamma(\mathbf{G}_2)$, $T_1 \in \gamma(?_\delta)$ and $T_2 \in \gamma(?_\delta)$. By the inductive hypothesis, we get that $\mathbf{G}_1 \sim ?_\delta$ and $\mathbf{G}_2 \sim ?_\delta$. Therefore, the result holds immediately, since $\mathbf{G}_1 \times \mathbf{G}_2 \sim ?_\delta$.

- $G = ?_\delta$ and $G' = G'_1 \times G'_2$. Identically to the previous case.
- $G = ?_\delta$ and $G' = ?_{\delta'}$. The result holds immediately, since $?_\delta \sim ?_{\delta'}$.

Case $(\forall X.T_1)$. Then, we know that $\forall X.T_1 \in \gamma(G)$ and $\forall X.T_1 \in \gamma(G')$. Thus, we have the following cases.

- $G = \forall X.G_1$ and $G' = \forall X.G'_1$. Thus, we know that $\forall X.T_1 \in \gamma(\forall X.G_1)$ and $\forall X.T_1 \in \gamma(\forall X.G'_1)$. Therefore, we get that $T_1 \in \gamma(G_1)$ and $T_1 \in \gamma(G'_1)$. By the inductive hypothesis, we get that $G_1 \sim G'_1$. Therefore, the result holds immediately, since $\forall X.G_1 \sim \forall X.G'_1$.
- $G = \forall X.G_1$ and $G' = ?_\delta$. Thus, we know that $\forall X.T_1 \in \gamma(\forall X.G_1)$ and $\forall X.T_1 \in \gamma(?_\delta) \supseteq \forall X.\gamma(?_\delta, X:X)$. Therefore, we get that $T_1 \in \gamma(G_1)$ and $T_1 \in \gamma(?_\delta, X:X)$. By the inductive hypothesis, we get that $G_1 \sim ?_\delta, X:X$. Therefore, the result holds immediately, since $\forall X.G_1 \sim ?_\delta$.
- $G = ?_\delta$ and $G' = \forall X.G'_1$. Identically to the previous case.
- $G = ?_\delta$ and $G' = ?_{\delta'}$. The result holds immediately, since $?_\delta \sim ?_{\delta'}$.

□

Definition B.97 (Consistent lifting of functions) *Let F_n be a function of type $\text{TYPE}^n \rightarrow \text{TYPE}$. Its consistent lifting F_n^\sharp , of type $\text{GTYPE}^n \rightarrow \text{GTYPE}$, is defined as: $F_n^\sharp(\overline{G}) = \alpha(\{F_n(\overline{T}) \mid \overline{T} \in \gamma(\overline{G})\})$*

The abstract interpretation framework allows us to prove the following definitions: [Consistent type functions] The definitions of dom^\sharp , cod^\sharp , inst^\sharp , and proj_i^\sharp given in Fig. B.6 are consistent liftings, as per Def. B.97, of the corresponding functions from Fig. B.3.

PROOF.

Case (dom^\sharp) . If $\text{dom}^\sharp(G) = \alpha(\{\text{dom}(T) \mid T \in \gamma(G)\})$ is defined, we know that $A = \{\text{dom}(T) \mid T \in \gamma(G)\} \neq \emptyset$. Therefore, by the definition of $\text{dom}(T)$, we know that at least $T = T_1 \rightarrow T_2 \in \gamma(G)$ and $T_1 \in A$. Therefore, we know that $G = ?_\delta$ or $G = G_1 \rightarrow G_2$.

- If $G = ?_\delta$, we know that $\text{dom}^\sharp(G) = \text{dom}^\sharp(?_\delta) = ?_\delta$ (definition in figure). We know that

$$\text{ftv}(\{\text{dom}(T) \mid T \in \gamma(?_\delta)\}) = \text{ftv}(\delta)$$

Therefore, we get that $\text{dom}^\sharp(G) = \text{dom}^\sharp(?_\delta) = \alpha(\{\text{dom}(T) \mid T \in \gamma(?_\delta)\}) = ?_\delta$ (Def. B.97). Thus, the result holds.

- If $G = G_1 \rightarrow G_2$, we know that $\text{dom}^\sharp(G) = \text{dom}^\sharp(G_1 \rightarrow G_2) = G_1$ (definition in figure). We know that $\gamma(G_1 \rightarrow G_2) = \gamma(G_1) \rightarrow \gamma(G_2) = \{T_1 \rightarrow T_2 \mid T_1 \in \gamma(G_1) \wedge T_2 \in \gamma(G_2)\}$. Also, we know that $\text{dom}^\sharp(G_1 \rightarrow G_2) = \alpha(\{\text{dom}(T) \mid T \in \gamma(G_1 \rightarrow G_2)\}) = \alpha(\{T_1 \mid T_1 \rightarrow T_2 \in \gamma(G_1) \rightarrow \gamma(G_2)\}) = \alpha(\gamma(G_1))$ (Def. B.97). By Corollary B.6.1, we know that $\alpha(\gamma(G_1)) = G_1$. Therefore, $\text{dom}^\sharp(G_1 \rightarrow G_2) = \alpha(\gamma(G_1)) = G_1$, and the result holds.

Case (cod^\sharp) . If $\text{cod}^\sharp(G) = \alpha(\{\text{cod}(T) \mid T \in \gamma(G)\})$ is defined, we know that $A = \{\text{cod}(T) \mid T \in \gamma(G)\} \neq \emptyset$. Therefore, by the definition of $\text{cod}(T)$, we know that at least $T = T_1 \rightarrow T_2 \in \gamma(G)$ and $T_1 \in A$. Therefore, we know that $G = ?_\delta$ or $G = G_1 \rightarrow G_2$.

- If $\mathbf{G} = ?_\delta$, we know that $\text{cod}^\sharp(\mathbf{G}) = \text{cod}^\sharp(?_\delta) = ?_\delta$ (definition in figure). We know that

$$\text{ftv}(\{ \text{cod}(T) \mid T \in \gamma(?_\delta) \}) = \text{ftv}(\delta)$$

Therefore, we get that $\text{cod}^\sharp(\mathbf{G}) = \text{cod}^\sharp(?_\delta) = \alpha(\{ \text{cod}(T) \mid T \in \gamma(?_\delta) \}) = ?_\delta$ (Def. B.97). Thus, the result holds.

- If $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$, we know that $\text{cod}^\sharp(\mathbf{G}) = \text{cod}^\sharp(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \mathbf{G}_2$ (definition in figure). We know that $\gamma(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}_2) = \{ T_1 \rightarrow T_2 \mid T_1 \in \gamma(\mathbf{G}_1) \wedge T_2 \in \gamma(\mathbf{G}_2) \}$. Also, we know that $\text{cod}^\sharp(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \alpha(\{ \text{cod}(T) \mid T \in \gamma(\mathbf{G}_1 \rightarrow \mathbf{G}_2) \}) = \alpha(\{ T_2 \mid T_1 \rightarrow T_2 \in \gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}_2) \}) = \alpha(\gamma(\mathbf{G}_2))$ (Def. B.97). By Corollary B.6.1, we know that $\alpha(\gamma(\mathbf{G}_2)) = \mathbf{G}_2$. Therefore, $\text{cod}^\sharp(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \alpha(\gamma(\mathbf{G}_2)) = \mathbf{G}_2$, and the result holds.

Case (inst[♯]). If $\text{inst}^\sharp(\mathbf{G}, \mathbf{G}') = \alpha(\{ \text{inst}(T, T') \mid T \in \gamma(\mathbf{G}) \wedge T' \in \gamma(\mathbf{G}') \})$ is defined, we know that

$$A = \{ \text{inst}(T, T') \mid T \in \gamma(\mathbf{G}) \wedge T' \in \gamma(\mathbf{G}') \} \neq \emptyset$$

Therefore, by the definition of $\text{inst}(T, T')$, we know that at least $T = \forall X.T_1 \in \gamma(\mathbf{G})$ and $T_1 \in A$. Therefore, we know that $\mathbf{G} = ?_\delta$ or $\mathbf{G} = \forall X.\mathbf{G}_1$.

- If $\mathbf{G} = ?_\delta$, we know that $\text{inst}^\sharp(\mathbf{G}, \mathbf{G}') = \text{inst}^\sharp(?_\delta, \mathbf{G}') = ?_\delta$ (definition in figure). We know that

$$\text{ftv}(\{ \text{inst}(T, T') \mid T \in \gamma(?_\delta) \wedge T' \in \gamma(\mathbf{G}') \}) = \text{ftv}(\delta)$$

Therefore, we get that $\text{inst}^\sharp(\mathbf{G}, \mathbf{G}') = \text{inst}^\sharp(?_\delta, \mathbf{G}') = \alpha(\{ \text{inst}(T, T') \mid T \in \gamma(?_\delta) \wedge T' \in \gamma(\mathbf{G}') \}) = ?_\delta$ (Def. B.97). Thus, the result holds.

- If $\mathbf{G} = \forall X.\mathbf{G}_1$, we know that $\text{inst}^\sharp(\mathbf{G}, \mathbf{G}') = \text{inst}^\sharp(\forall X.\mathbf{G}_1, \mathbf{G}') = \mathbf{G}_1[\mathbf{G}'/X] \setminus X$ (definition in figure). We know that $\gamma(\forall X.\mathbf{G}_1) = \forall X.\gamma(\mathbf{G}_1) = \{ \forall X.T_1 \mid T_1 \in \gamma(\mathbf{G}_1) \}$. Also, we know that

$$\text{inst}^\sharp(\forall X.\mathbf{G}_1, \mathbf{G}') = \alpha(\{ \text{inst}(T, T') \mid T \in \gamma(\forall X.\mathbf{G}_1) \wedge T' \in \gamma(\mathbf{G}') \}) =$$

$$\alpha(\{ T_1[T'/X] \setminus X \mid \forall X.T_1 \in \forall X.\gamma(\mathbf{G}_1) \wedge T' \in \gamma(\mathbf{G}') \}) = \alpha(\gamma(\mathbf{G}_1[\mathbf{G}'/X] \setminus X))$$

(Def. B.97). By Corollary B.6.1, we know that $\alpha(\gamma(\mathbf{G}_1[\mathbf{G}'/X] \setminus X)) = \mathbf{G}_1[\mathbf{G}'/X] \setminus X$. Therefore, $\text{inst}^\sharp(\forall X.\mathbf{G}_1, \mathbf{G}') = \alpha(\gamma(\mathbf{G}_1[\mathbf{G}'/X] \setminus X)) = \mathbf{G}_1[\mathbf{G}'/X] \setminus X$, and the result holds.

Case (proj_i[♯]). If $\text{proj}_i^\sharp(\mathbf{G}) = \alpha(\{ \text{proj}_i(T) \mid T \in \gamma(\mathbf{G}) \})$ is defined, we know that $A = \{ \text{proj}_i(T) \mid T \in \gamma(\mathbf{G}) \} \neq \emptyset$. Therefore, by the definition of $\text{proj}_i(T)$, we know that at least $T = T_1 \times T_2 \in \gamma(\mathbf{G})$ and $T_1 \in A$. Therefore, we know that $\mathbf{G} = ?_\delta$ or $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$.

- If $\mathbf{G} = ?_\delta$, we know that $\text{proj}_i^\sharp(\mathbf{G}) = \text{proj}_i^\sharp(?_\delta) = ?_\delta$ (definition in figure). We know that

$$\text{ftv}(\{ \text{proj}_i(T) \mid T \in \gamma(?_\delta) \}) = \text{ftv}(\delta)$$

Therefore, we get that $\text{proj}_i^\sharp(\mathbf{G}) = \text{proj}_i^\sharp(?_\delta) = \alpha(\{ \text{proj}_i(T) \mid T \in \gamma(?_\delta) \}) = ?_\delta$ (Def. B.97). Thus, the result holds.

- If $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$, we know that $\text{proj}_i^\sharp(\mathbf{G}) = \text{proj}_i^\sharp(\mathbf{G}_1 \times \mathbf{G}_2) = \mathbf{G}_i$ (definition in figure). We know that $\gamma(\mathbf{G}_1 \times \mathbf{G}_2) = \gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}_2) = \{ T_1 \times T_2 \mid T_1 \in \gamma(\mathbf{G}_1) \wedge T_2 \in \gamma(\mathbf{G}_2) \}$. Also, we know that $\text{proj}_i^\sharp(\mathbf{G}_1 \times \mathbf{G}_2) = \alpha(\{ \text{proj}_i(T) \mid T \in \gamma(\mathbf{G}_1 \times \mathbf{G}_2) \}) = \alpha(\{ T_i \mid T_1 \times T_2 \in \gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}_2) \}) = \alpha(\gamma(\mathbf{G}_i))$ (Def. B.97). By Corollary B.6.1, we know that $\alpha(\gamma(\mathbf{G}_i)) = \mathbf{G}_i$. Therefore, $\text{proj}_i^\sharp(\mathbf{G}_1 \times \mathbf{G}_2) = \alpha(\gamma(\mathbf{G}_i)) = \mathbf{G}_i$, and the result holds.

□

Lemma B.98 *If $A \subseteq \gamma(\mathbf{G})$ then $\text{ftv}(A) \subseteq \text{ftv}(\mathbf{G})$.*

PROOF. We proceed by induction on \mathbf{G} .

Case (B). We know that $\gamma(\mathbf{B}) = \{\mathbf{B}\}$. Since A is no empty and $A \subseteq \gamma(\mathbf{G})$, then we know that $A = \gamma(\mathbf{B}) = \{\mathbf{B}\}$. Also, we know that $\text{ftv}(\{\mathbf{B}\}) = \emptyset \subseteq \emptyset = \text{ftv}(\mathbf{B})$, as we are required to prove.

Case (X). We know that $\gamma(\mathbf{X}) = \{\mathbf{X}\}$. Since A is no empty and $A \subseteq \gamma(\mathbf{G})$, then we know that $A = \gamma(\mathbf{X}) = \{\mathbf{X}\}$. Also, we know that $\text{ftv}(\{\mathbf{X}\}) = \{\mathbf{X}\} \subseteq \{\mathbf{X}\} = \text{ftv}(\mathbf{X})$, as we are required to prove.

Case ($\mathbf{G}_1 \rightarrow \mathbf{G}_2$). We know that $\gamma(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}_2)$ and $\text{ftv}(\gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}_2)) = \text{ftv}(\gamma(\mathbf{G}_1)) \cup \text{ftv}(\gamma(\mathbf{G}_2))$. Since A is no empty and $A \subseteq \gamma(\mathbf{G})$, then we know that $A = \{\overline{T_{11} \rightarrow T_{12}}\}$, $\text{ftv}(\{\overline{T_{11} \rightarrow T_{12}}\}) = \text{ftv}(\{\overline{T_{11}}\}) \cup \text{ftv}(\{\overline{T_{12}}\})$, $\{\overline{T_{11}}\} \subseteq \gamma(\mathbf{G}_1)$ and $\{\overline{T_{12}}\} \subseteq \gamma(\mathbf{G}_2)$. Therefore, we are required to show that $\text{ftv}(\{\overline{T_{11}}\}) \cup \text{ftv}(\{\overline{T_{12}}\}) \subseteq \text{ftv}(\gamma(\mathbf{G}_1)) \cup \text{ftv}(\gamma(\mathbf{G}_2))$. By the inductive hypothesis on \mathbf{G}_1 and \mathbf{G}_2 , with $\{\overline{T_{11}}\} \subseteq \gamma(\mathbf{G}_1)$ and $\{\overline{T_{12}}\} \subseteq \gamma(\mathbf{G}_2)$, we get that $\text{ftv}(\{\overline{T_{11}}\}) \subseteq \text{ftv}(\gamma(\mathbf{G}_1))$ and $\text{ftv}(\{\overline{T_{12}}\}) \subseteq \text{ftv}(\gamma(\mathbf{G}_2))$. Thus, we have that $\text{ftv}(\{\overline{T_{11}}\}) \cup \text{ftv}(\{\overline{T_{12}}\}) \subseteq \text{ftv}(\gamma(\mathbf{G}_1)) \cup \text{ftv}(\gamma(\mathbf{G}_2))$, as we required to prove.

Case ($\mathbf{G}_1 \times \mathbf{G}_2$). We know that $\gamma(\mathbf{G}_1 \times \mathbf{G}_2) = \gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}_2)$ and $\text{ftv}(\gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}_2)) = \text{ftv}(\gamma(\mathbf{G}_1)) \cup \text{ftv}(\gamma(\mathbf{G}_2))$. Since A is no empty and $A \subseteq \gamma(\mathbf{G})$, then we know that $A = \{\overline{T_{11} \times T_{12}}\}$, $\text{ftv}(\{\overline{T_{11} \times T_{12}}\}) = \text{ftv}(\{\overline{T_{11}}\}) \cup \text{ftv}(\{\overline{T_{12}}\})$, $\{\overline{T_{11}}\} \subseteq \gamma(\mathbf{G}_1)$ and $\{\overline{T_{12}}\} \subseteq \gamma(\mathbf{G}_2)$. Therefore, we are required to show that $\text{ftv}(\{\overline{T_{11}}\}) \cup \text{ftv}(\{\overline{T_{12}}\}) \subseteq \text{ftv}(\gamma(\mathbf{G}_1)) \cup \text{ftv}(\gamma(\mathbf{G}_2))$. By the inductive hypothesis on \mathbf{G}_1 and \mathbf{G}_2 , with $\{\overline{T_{11}}\} \subseteq \gamma(\mathbf{G}_1)$ and $\{\overline{T_{12}}\} \subseteq \gamma(\mathbf{G}_2)$, we get that $\text{ftv}(\{\overline{T_{11}}\}) \subseteq \text{ftv}(\gamma(\mathbf{G}_1))$ and $\text{ftv}(\{\overline{T_{12}}\}) \subseteq \text{ftv}(\gamma(\mathbf{G}_2))$. Thus, we have that $\text{ftv}(\{\overline{T_{11}}\}) \cup \text{ftv}(\{\overline{T_{12}}\}) \subseteq \text{ftv}(\gamma(\mathbf{G}_1)) \cup \text{ftv}(\gamma(\mathbf{G}_2))$, as we required to prove.

Case ($\forall \mathbf{X}.\mathbf{G}_1$). We know that $\gamma(\forall \mathbf{X}.\mathbf{G}_1) = \forall \mathbf{X}.\gamma(\mathbf{G}_1)$ and $\text{ftv}(\forall \mathbf{X}.\gamma(\mathbf{G}_1)) = \text{ftv}(\gamma(\mathbf{G}_1)) \setminus \mathbf{X}$. Since A is no empty and $A \subseteq \gamma(\mathbf{G})$, then we know that $A = \{\overline{\forall \mathbf{X}.T_{11}}\}$, $\text{ftv}(\{\overline{\forall \mathbf{X}.T_{11}}\}) = \text{ftv}(\{\overline{T_{11}}\}) \setminus \mathbf{X}$, $\{\overline{T_{11}}\} \subseteq \gamma(\mathbf{G}_1)$. Therefore, we are required to show that $\text{ftv}(\{\overline{T_{11}}\}) \setminus \mathbf{X} \subseteq \text{ftv}(\gamma(\mathbf{G}_1)) \setminus \mathbf{X}$. By the inductive hypothesis on \mathbf{G}_1 , with $\{\overline{T_{11}}\} \subseteq \gamma(\mathbf{G}_1)$, we get that $\text{ftv}(\{\overline{T_{11}}\}) \subseteq \text{ftv}(\gamma(\mathbf{G}_1))$. Thus, we have that $\text{ftv}(\{\overline{T_{11}}\}) \setminus \mathbf{X} \subseteq \text{ftv}(\gamma(\mathbf{G}_1)) \setminus \mathbf{X}$, as we required to prove.

Case ($?_\delta$). We know that $\gamma(?_\delta) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. Since $A \subseteq \gamma(?_\delta)$, we know that $\text{ftv}(A) \subseteq \text{ftv}(\gamma(?_\delta)) = \text{ftv}(\gamma(?_\delta)) = \text{ftv}(\delta)$, by Lemma B.101. Therefore, since $A \subseteq \gamma(?_\delta)$, we know that $\forall T \in A, \text{ftv}(T) \subseteq \text{ftv}(\delta) = \text{ftv}(\gamma(?_\delta))$, therefore, we get that $\text{ftv}(A) \subseteq \text{ftv}(\gamma(?_\delta))$, as we are required to prove.

□

Lemma B.99 *If $\mathbf{G} \sqsubseteq \mathbf{G}'$ then $\text{ftv}(\mathbf{G}) \subseteq \text{ftv}(\mathbf{G}')$.*

PROOF. We know that $\mathbf{G} \sqsubseteq \mathbf{G}'$ and we are required to show that $\text{ftv}(\mathbf{G}) \subseteq \text{ftv}(\mathbf{G}')$. Since $\mathbf{G} \sqsubseteq \mathbf{G}'$, by Definition B.94, we know that $\gamma(\mathbf{G}) \subseteq \gamma(\mathbf{G}')$. Therefore, $\text{ftv}(\gamma(\mathbf{G})) \subseteq \text{ftv}(\gamma(\mathbf{G}'))$. Thus, by Lemma B.101, we know that $\text{ftv}(\gamma(\mathbf{G})) = \text{ftv}(\mathbf{G})$ and $\text{ftv}(\gamma(\mathbf{G}')) = \text{ftv}(\mathbf{G}')$. Then, we get that $\text{ftv}(\gamma(\mathbf{G})) = \text{ftv}(\mathbf{G}) \subseteq \text{ftv}(\mathbf{G}') = \text{ftv}(\gamma(\mathbf{G}'))$, as we are required to show. □

Lemma B.100 *If $\text{ftv}(\mathbf{G}) \subseteq \text{ftv}(\gamma(?_\delta))$ then $\mathbf{G} \sqsubseteq ?_\delta$.*

PROOF. We know that $\text{ftv}(\mathbf{G}) \subseteq \text{ftv}(\text{?}_\delta)$ and we are required to show that $\mathbf{G} \sqsubseteq \text{?}_\delta$. By Definition B.94, we are required to prove that $\gamma(\mathbf{G}) \subseteq \gamma(\text{?}_\delta)$. We know that $\gamma(\text{?}_\delta) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. By Lemma B.101, we know that $\text{ftv}(\mathbf{G}) = \text{ftv}(\gamma(\mathbf{G}))$. Since $\text{ftv}(\gamma(\mathbf{G})) = \text{ftv}(\mathbf{G}) \subseteq \text{ftv}(\text{?}_\delta) = \text{ftv}(\delta)$, we know that $\forall T \in \gamma(\mathbf{G}), \text{ftv}(T) \subseteq \text{ftv}(\gamma(\mathbf{G})) \subseteq \text{ftv}(\delta)$, therefore, we get that $\forall T \in \gamma(\mathbf{G}), T \in \gamma(\text{?}_\delta)$. Thus, $\gamma(\mathbf{G}) \subseteq \gamma(\text{?}_\delta)$, as we are required to prove. \square

Lemma B.101 $\text{ftv}(\mathbf{G}) = \text{ftv}(\gamma(\mathbf{G}))$.

PROOF. We are required to prove that $\text{ftv}(\mathbf{G}) = \text{ftv}(\gamma(\mathbf{G}))$. We proceed by induction on \mathbf{G} .

Case (B). We know that $\text{ftv}(\mathbf{G}) = \text{ftv}(\mathbf{B}) = \emptyset = \text{ftv}(\{\mathbf{B}\}) = \text{ftv}(\gamma(\mathbf{B})) = \text{ftv}(\gamma(\mathbf{G}))$. Therefore, the result follows immediately.

Case (X). We know that $\text{ftv}(\mathbf{G}) = \text{ftv}(\mathbf{X}) = \{\mathbf{X}\} = \text{ftv}(\{\mathbf{X}\}) = \text{ftv}(\gamma(\mathbf{X})) = \text{ftv}(\gamma(\mathbf{G}))$. Therefore, the result follows immediately.

Case ($\mathbf{G}_1 \rightarrow \mathbf{G}_2$). We know that $\text{ftv}(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \text{ftv}(\mathbf{G}_1) \cup \text{ftv}(\mathbf{G}_2)$ and $\gamma(\mathbf{G}_1 \rightarrow \mathbf{G}_2) = \gamma(\mathbf{G}_1) \rightarrow \gamma(\mathbf{G}_2)$. Therefore, we have that $\text{ftv}(\gamma(\mathbf{G}_1 \rightarrow \mathbf{G}_2)) = \text{ftv}(\gamma(\mathbf{G}_1)) \cup \text{ftv}(\gamma(\mathbf{G}_2))$. By the inductive hypothesis on \mathbf{G}_1 and \mathbf{G}_2 , we get that $\text{ftv}(\mathbf{G}_1) = \text{ftv}(\gamma(\mathbf{G}_1))$ and $\text{ftv}(\mathbf{G}_2) = \text{ftv}(\gamma(\mathbf{G}_2))$. Thus, we get that $\text{ftv}(\mathbf{G}_1) \cup \text{ftv}(\mathbf{G}_2) = \text{ftv}(\gamma(\mathbf{G}_1)) \cup \text{ftv}(\gamma(\mathbf{G}_2))$, as we are required to prove.

Case ($\mathbf{G}_1 \times \mathbf{G}_2$). We know that $\text{ftv}(\mathbf{G}_1 \times \mathbf{G}_2) = \text{ftv}(\mathbf{G}_1) \cup \text{ftv}(\mathbf{G}_2)$ and $\gamma(\mathbf{G}_1 \times \mathbf{G}_2) = \gamma(\mathbf{G}_1) \times \gamma(\mathbf{G}_2)$. Therefore, we have that $\text{ftv}(\gamma(\mathbf{G}_1 \times \mathbf{G}_2)) = \text{ftv}(\gamma(\mathbf{G}_1)) \cup \text{ftv}(\gamma(\mathbf{G}_2))$. By the inductive hypothesis on \mathbf{G}_1 and \mathbf{G}_2 , we get that $\text{ftv}(\mathbf{G}_1) = \text{ftv}(\gamma(\mathbf{G}_1))$ and $\text{ftv}(\mathbf{G}_2) = \text{ftv}(\gamma(\mathbf{G}_2))$. Thus, we get that $\text{ftv}(\mathbf{G}_1) \cup \text{ftv}(\mathbf{G}_2) = \text{ftv}(\gamma(\mathbf{G}_1)) \cup \text{ftv}(\gamma(\mathbf{G}_2))$, as we are required to prove.

Case ($\forall \mathbf{X}.\mathbf{G}_1$). We know that $\text{ftv}(\forall \mathbf{X}.\mathbf{G}_1) = \text{ftv}(\mathbf{G}_1) \setminus \{\mathbf{X}\}$ and $\gamma(\forall \mathbf{X}.\mathbf{G}_1) = \forall \mathbf{X}.\gamma(\mathbf{G}_1)$. Therefore, we have that $\text{ftv}(\gamma(\forall \mathbf{X}.\mathbf{G}_1)) = \text{ftv}(\gamma(\mathbf{G}_1)) \setminus \{\mathbf{X}\}$. By the inductive hypothesis on \mathbf{G}_1 , we get that $\text{ftv}(\mathbf{G}_1) = \text{ftv}(\gamma(\mathbf{G}_1))$. Thus, we get that $\text{ftv}(\mathbf{G}_1) \setminus \{\mathbf{X}\} = \text{ftv}(\gamma(\mathbf{G}_1)) \setminus \{\mathbf{X}\}$, as we are required to prove.

Case (?_δ). We know that $\text{ftv}(\text{?}_\delta) = \text{ftv}(\delta)$ and $\gamma(\text{?}_\delta) = \{T \mid \text{ftv}(T) \subseteq \text{ftv}(\delta)\}$. Therefore, we have that $\text{ftv}(\gamma(\text{?}_\delta)) = \text{ftv}(\delta)$, as we are required to prove. \square

Lemma B.102 $\mathbf{G} \sqcap \mathbf{G} = \mathbf{G}$.

PROOF. We know that $\mathbf{G} \sqcap \mathbf{G} = \alpha(\gamma(\mathbf{G}) \cap \gamma(\mathbf{G})) = \alpha(\gamma(\mathbf{G})) = \mathbf{G}$, by Corollary B.6.1. \square

Lemma B.103 *If $\mathbf{G}_1 \sim \mathbf{G}_2$ then $\mathbf{G}_1 \sqcap \mathbf{G}_2$ is defined.*

PROOF. Since, $\mathbf{G}_1 \sqcap \mathbf{G}_2 = \alpha(\gamma(\mathbf{G}_1) \cap \gamma(\mathbf{G}_2))$, by Definition B.96, we know that $\gamma(\mathbf{G}_1) \cap \gamma(\mathbf{G}_2) \neq \emptyset$. Therefore, we have that $\alpha(\gamma(\mathbf{G}_1) \cap \gamma(\mathbf{G}_2))$, is defined, and the result holds. \square

Lemma B.104 *If $\mathbf{G}_1 \sim \mathbf{G}_2$ then $\text{initEv}(\mathbf{G}_1, \mathbf{G}_2)$.*

PROOF. We know that $\text{initEv}(\mathbf{G}_1, \mathbf{G}_2) = \{(\mathbf{G}_1 \sqcap \mathbf{G}_2, \text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_1), \text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_2))\}$. By Lemma B.103, we have that $\mathbf{G}_1 \sqcap \mathbf{G}_2$ is defined. We know by Lemma B.118, that $\mathbf{G}_1 \sqcap \mathbf{G}_2 \sqsubseteq \mathbf{G}_1$ and $\mathbf{G}_1 \sqcap \mathbf{G}_2 \sqsubseteq \mathbf{G}_2$. Since $\mathbf{G}_1 \sqcap \mathbf{G}_2 \sqsubseteq \mathbf{G}_1$ and $\mathbf{G}_1 \sqcap \mathbf{G}_2 \sqsubseteq \mathbf{G}_2$, we know that $\text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_1)$ and $\text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_2)$, are defined. Thus, $\text{initEv}(\mathbf{G}_1, \mathbf{G}_2)$ is define, and the result follows. \square

Lemma B.105 *If $G_1 \sqsubseteq G_2$ then $\text{initPT}(G_1, G_2)$.*

PROOF. Straightforward, by Definition of $\text{initPT}(G_1, G_2)$ (Definition B.106). \square

B.6.2 Typing judgement

$$\begin{aligned}
?_{\delta_1; X; \delta_2}[G'/X] \setminus X &= ?_{\delta_1; \delta_2} \\
B[G'/X] \setminus X &= B \\
X[G'/X] \setminus X &= G' \\
Y[G'/X] \setminus X &= Y \\
G_1 \rightarrow G_2[G'/X] \setminus X &= G_1[G'/X] \setminus X \rightarrow G_2[G'/X] \setminus X \\
G_1 \times G_2[G'/X] \setminus X &= G_1[G'/X] \setminus X \times G_2[G'/X] \setminus X \\
\forall Y. G[G'/X] \setminus X &= \forall Y. G[G'/X] \setminus X
\end{aligned}$$

B.6.3 Elaborating $F^?$ to $F^?_\varepsilon$

Definition B.106 (Initial Proof) *If $G \sqsubseteq G'$, then we define $\text{initPT}(G, G') : G \sqsubseteq G'$ as follows:*

$$\begin{aligned}
\text{initPT}(B, B) &= \text{refl}_B \\
\text{initPT}(X, X) &= \text{refl}_X \\
\text{initPT}(G_1 \rightarrow G_2, G'_1 \rightarrow G'_2) &= \text{initPT}(G_1, G'_1) \rightarrow \text{initPT}(G_2, G'_2) \\
\text{initPT}(G_1 \times G_2, G'_1 \times G'_2) &= \text{initPT}(G_1, G'_1) \times \text{initPT}(G_2, G'_2) \\
\text{initPT}(\forall X. G_1, \forall X. G'_1) &= \forall X. \text{initPT}(G_1, G'_1) \\
\text{initPT}(B, ?_\delta) &= \text{inj}_B \\
\text{initPT}(X, ?_\delta) &= \text{inj}_X \\
\text{initPT}(G_1 \rightarrow G_2, ?_\delta) &= \text{inj}_{\rightarrow}(\text{initPT}(G_1 \rightarrow G_2, ?_\delta \rightarrow ?_\delta)) \\
\text{initPT}(G_1 \times G_2, ?_\delta) &= \text{inj}_{\times}(\text{initPT}(G_1 \times G_2, ?_\delta \times ?_\delta)) \\
\text{initPT}(\forall X. G_1, ?_\delta) &= \text{inj}_{\forall}(\text{initPT}(\forall X. G_1, \forall X. ?_\delta, X : X)) \\
\text{initPT}(?_\delta, ?_{\delta'}) &= \text{inj}_?
\end{aligned}$$

Example of elaboration. We now present the elaboration of program 2 of Table 6.1. Program $(\lambda X. \lambda x : ?_X :: X)$ [Bool] 42 is elaborated as follows.

1. As $x : ?_X$, then $x :: X$ elaborates to $t_1 = \varepsilon_1 x :: X$, where $\varepsilon_1 = \{(X, \text{inj}_X, X)\}$ justifies that $?_X \sim X$.
2. As all values in $F^?_\varepsilon$ are ascribed, $\lambda x : ?_X. x :: X$ is elaborated to $t_2 = \varepsilon_2(\lambda x : ?_X. t_1) :: ?_X \rightarrow X$, where $\varepsilon_2 = \{(?_X \rightarrow X, \text{inj}_{\rightarrow} \rightarrow \text{refl}_X, \text{inj}_{\rightarrow} \rightarrow \text{refl}_X)\}$ justifies that $?_X \rightarrow X \sim ?_X \rightarrow X$.
3. Similarly, the type abstraction is also ascribed: $\lambda X. \lambda x : ?_X. x :: X$ elaborates to $t_3 = \varepsilon_3(\lambda X. t_2) :: \forall X. ?_X \rightarrow X$, where $\varepsilon_3 = \{(\forall X. ?_X \rightarrow X, \forall X. \text{inj}_{\rightarrow} \rightarrow \text{refl}_X, \forall X. \text{inj}_{\rightarrow} \rightarrow \text{refl}_X)\}$ justifies that $\forall X. ?_X \rightarrow X \sim \forall X. ?_X \rightarrow X$.

4. Type application $(\lambda X. \lambda x : ?_X. x :: X) [\mathbf{Bool}]$ elaborates to $t_4 = \varepsilon_4(t_3 [\mathbf{Bool}]) :: ? \rightarrow \mathbf{Bool}$, where
 $\varepsilon_4 = \{(\mathbf{Bool} \rightarrow \mathbf{Bool}, \text{inj}_X \rightarrow \text{refl}_{\mathbf{Bool}}, \text{inj}_{\mathbf{Bool}} \rightarrow \mathbf{Bool}), (? \rightarrow \mathbf{Bool}, \text{inj}_? \rightarrow \text{refl}_{\mathbf{Bool}}, \text{inj}_? \rightarrow \text{refl}_{\mathbf{Bool}})\}$
justifies that $?_X \rightarrow \mathbf{Bool} \sim ? \rightarrow \mathbf{Bool}$ (sealing/unsealing evidence).
5. Number 42 is ascribed to its type and elaborated to $t_5 = \varepsilon_5 42 :: \mathbf{Int}$, where $\varepsilon_5 = \{(\mathbf{Int}, \text{refl}_{\mathbf{Int}}, \mathbf{Int})\}$
which justifies the reflexive judgement $\mathbf{Int} \sim \mathbf{Int}$.
6. The elaborated function argument t_5 is ascribed to the domain of the function type: $t_6 = \varepsilon_6 t_5 :: ?$, where $\varepsilon_6 = \{(\mathbf{Int}, \text{refl}_{\mathbf{Int}}, \text{inj}_{\mathbf{Int}})\}$ which justifies that $\mathbf{Int} \sim ?$.
7. Finally, the whole program elaborates to
 $t_4 t_6 = (\varepsilon_4(\varepsilon_3(\lambda X. \varepsilon_2(\lambda x : ?_X. \varepsilon_1 x :: X) :: ?_X \rightarrow X) :: \forall X. ?_X \rightarrow X [\mathbf{Bool}]) :: ? \rightarrow \mathbf{Bool}) (\varepsilon_6(\varepsilon_5 42 :: \mathbf{Int}) :: ?)$.

Armed with the dynamic semantics of $F_\varepsilon^?$ and the concrete representation of evidence, we now illustrate the reduction of program 2 of Table 6.1: $(\lambda X. \lambda x : ?_X. x :: X) [\mathbf{Bool}] 42$. Its complete elaboration is the $F_\varepsilon^?$ term $(\varepsilon_4(\varepsilon_3(\lambda X. \varepsilon_2(\lambda x : ?_X. \varepsilon_1 x :: X) :: ?_X \rightarrow X) :: \forall X. ?_X \rightarrow X [\mathbf{Bool}]) :: ? \rightarrow \mathbf{Bool}) (\varepsilon_6(\varepsilon_5 42 :: \mathbf{Int}) :: ?)$, where $\varepsilon_1 = \{(X, \text{inj}_X, \text{refl}_X)\}$, $\varepsilon_2 = \{(?_X \rightarrow X, \text{inj}_? \rightarrow \text{refl}_X, \text{inj}_? \rightarrow \text{refl}_X)\}$, $\varepsilon_3 = \{(\forall X. ?_X \rightarrow X, \forall X. \text{inj}_? \rightarrow \text{refl}_X, \forall X. \text{inj}_? \rightarrow \text{refl}_X)\}$, $\varepsilon_4 = \{(\mathbf{Bool} \rightarrow \mathbf{Bool}, \text{inj}_X \rightarrow \text{refl}_{\mathbf{Bool}}, \text{inj}_{\mathbf{Bool}} \rightarrow \text{refl}_{\mathbf{Bool}}), (? \rightarrow \mathbf{Bool}, \text{inj}_? \rightarrow \text{refl}_{\mathbf{Bool}}, \text{inj}_? \rightarrow \text{refl}_{\mathbf{Bool}})\}$, $\varepsilon_5 = \{(\mathbf{Int}, \text{refl}_{\mathbf{Int}}, \text{refl}_{\mathbf{Int}})\}$, and $\varepsilon_6 = \{(\mathbf{Int}, \text{refl}_{\mathbf{Int}}, \text{inj}_{\mathbf{Int}})\}$. The program reduces as follows:

$$\begin{aligned}
& (\varepsilon_4(\varepsilon_3(\lambda X. \varepsilon_2(\lambda x : ?_X. \varepsilon_1 x :: X) :: ?_X \rightarrow X) :: \forall X. ?_X \rightarrow X [\mathbf{Bool}]) :: ? \rightarrow \mathbf{Bool}) (\varepsilon_6(\varepsilon_5 42 :: \mathbf{Int}) :: ?) \\
\text{(RappG)} \quad & \rightarrow \varepsilon_4(\varepsilon'_3(\varepsilon'_2(\lambda x : ?_{X:\mathbf{Bool}}. \varepsilon'_1 x :: \mathbf{Bool}) :: ?_{X:\mathbf{Bool}} \rightarrow \mathbf{Bool}) :: ?_{X:\mathbf{Bool}} \rightarrow \mathbf{Bool}) (\varepsilon_6(\varepsilon_5 42 :: \mathbf{Int}) :: ?) \\
& \quad \text{where } \varepsilon'_1 = \{(\mathbf{Bool}, \text{inj}_X, \text{refl}_{\mathbf{Bool}})\} \text{ and } \varepsilon'_2 = \varepsilon'_3 = \{(?_{X:\mathbf{Bool}} \rightarrow \mathbf{Bool}, \text{inj}_? \rightarrow \text{refl}_{\mathbf{Bool}}, \text{inj}_? \rightarrow \text{refl}_{\mathbf{Bool}})\} \\
\text{(Rasc)} \quad & \rightarrow \varepsilon_4(\varepsilon'_2(\lambda x : ?_{X:\mathbf{Bool}}. \varepsilon'_1 x :: \mathbf{Bool}) :: ?_{X:\mathbf{Bool}} \rightarrow \mathbf{Bool}) (\varepsilon_6(\varepsilon_5 42 :: \mathbf{Int}) :: ?) \\
\text{(Rasc)} \quad & \rightarrow \varepsilon_4(\lambda x : ?_{X:\mathbf{Bool}}. \varepsilon'_1 x :: \mathbf{Bool}) (\varepsilon_6(\varepsilon_5 42 :: \mathbf{Int}) :: ?) \\
& \quad \text{because } \varepsilon'_3 \circ \varepsilon_4 = \varepsilon_4 \\
\text{(Rasc)} \quad & \rightarrow \varepsilon_4(\lambda x : ?_{X:\mathbf{Bool}}. \varepsilon'_1 x :: \mathbf{Bool}) (\varepsilon_6 42 :: \mathbf{Int}) \\
& \quad \text{because } \varepsilon'_5 \circ \varepsilon_6 = \varepsilon_6 \\
\text{(Rapp)} \quad & \rightarrow \varepsilon_5(\varepsilon'_1(\varepsilon 42 :: ?) :: ?_{X:\mathbf{Int}}) :: \mathbf{Bool} \\
& \quad \text{where } \varepsilon \circ \text{dom}(\varepsilon_4) = \varepsilon \text{ and } \varepsilon_5 = \text{cod}(\varepsilon_4) = \{(\mathbf{Bool}, \text{refl}_{\mathbf{Bool}}, \text{refl}_{\mathbf{Bool}})\} \\
\text{(Rasc)} \quad & \rightarrow \mathbf{error} \\
& \quad \text{because } \{(\mathbf{Int}, \text{refl}_{\mathbf{Int}}, \text{inj}_{\mathbf{Int}})\} \circ \{(\mathbf{Bool}, \text{inj}_X, \text{refl}_{\mathbf{Bool}})\} \text{ is undefined}
\end{aligned}$$

Theorem B.107 (Elaboration Preserves Typing) *If $\Delta; \Gamma \vdash t : G$, then $\Delta; \Gamma \vdash t : G \rightsquigarrow t'$ and $\Delta; \Gamma \vdash t' : G$.*

PROOF. The proof proceed by induction on the typing derivation of $\Delta; \Gamma \vdash t : G$.

Case (Gx). We know that $t = x$ and

$$\text{(Gx)} \frac{x : G \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x : G}$$

Thus, we get that

$$\text{(EGx)} \frac{x : G \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x : G \rightsquigarrow x}$$

Therefore, by (Gx) rule, we get that $\Delta; \Gamma \vdash x : G$, and the result follows immediately.

Case (Gb). We know that $t = b$, $G = B$, and

$$(Gb) \frac{\theta(b) = B \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash b : B}$$

Thus, we get that

$$(EGb) \frac{\theta(b) = B \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash b : B \rightsquigarrow \text{reflEv}(B) b :: B}$$

Since $\text{reflEv}(B) : B \sim B$, by (Gb) and (Gasc) rules, we get that $\Delta; \Gamma \vdash \text{reflEv}(B) b :: B : B$, and the result follows immediately.

Case (Gλ). We know that $t = \lambda x : G_1.t_1$, $G = G_1 \rightarrow G_2$, and

$$(G\lambda) \frac{\Delta; \Gamma, x : G_1 \vdash t_1 : G_2}{\Delta; \Gamma \vdash \lambda x : G_1.t_1 : G_1 \rightarrow G_2}$$

Since $\Delta; \Gamma, x : G_1 \vdash t_1 : G_2$, by the induction hypothesis, we get that $\Delta; \Gamma, x : G_1 \vdash t_1 : G_2 \rightsquigarrow t'_1$ and $\Delta; \Gamma, x : G_1 \vdash t'_1 : G_2$. Thus, we get that

$$(EG\lambda) \frac{\Delta; \Gamma, x : G_1 \vdash t_1 : G_2 \rightsquigarrow t'_1}{\Delta; \Gamma \vdash \lambda x : G_1.t_1 : G_1 \rightarrow G_2 \rightsquigarrow \text{reflEv}(G_1 \rightarrow G_2) (\lambda x : G_1.t'_1) :: G_1 \rightarrow G_2}$$

Since $\text{reflEv}(G_1 \rightarrow G_2) : G_1 \rightarrow G_2 \sim G_1 \rightarrow G_2$, by (Gλ) and (Gasc) rules, we get that $\Delta; \Gamma \vdash \text{reflEv}(G_1 \rightarrow G_2) (\lambda x : G_1.t'_1) :: G_1 \rightarrow G_2 : G_1 \rightarrow G_2$, and the result follows immediately.

Case (Gpair). We know that $t = \langle t_1, t_2 \rangle$, $G = G_1 \times G_2$, and

$$(G\text{pair}) \frac{\Delta; \Gamma \vdash t_1 : G_1 \quad \Delta; \Gamma \vdash t_2 : G_2}{\Delta; \Gamma \vdash \langle t_1, t_2 \rangle : G_1 \times G_2}$$

Since $\Delta; \Gamma \vdash t_1 : G_1$ and $\Delta; \Gamma \vdash t_2 : G_2$, by the induction hypothesis, we get that $\Delta; \Gamma \vdash t_1 : G_1 \rightsquigarrow t'_1$, $\Delta; \Gamma \vdash t_1 : G_1$, $\Delta; \Gamma \vdash t_2 : G_2 \rightsquigarrow t'_2$ and $\Delta; \Gamma \vdash t_2 : G_2$. Thus, we get that

$$(EG\text{pair}) \frac{\Delta; \Gamma \vdash t_1 : G_1 \rightsquigarrow t'_1 \quad \Delta; \Gamma \vdash t_2 : G_2 \rightsquigarrow t'_2}{\Delta; \Gamma \vdash \langle t_1, t_2 \rangle : G_1 \times G_2 \rightsquigarrow \langle t'_1, t'_2 \rangle}$$

By (Gpair) rule, we get that $\Delta; \Gamma \vdash \langle t'_1, t'_2 \rangle : G_1 \times G_2$, and the result follows immediately.

Case (GΛ). We know that $t = \Lambda X.t_1$, $G = \forall X.G_1$, and

$$(G\Lambda) \frac{\Delta, X; \Gamma \vdash t_1 : G_1 \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \Lambda X.t_1 : \forall X.G_1}$$

Since $\Delta, X; \Gamma \vdash t_1 : G_1$, by the induction hypothesis, we get that $\Delta, X; \Gamma \vdash t_1 : G_1 \rightsquigarrow t'_1$ and $\Delta, X; \Gamma \vdash t'_1 : G_1$. Thus, we get that

$$(EG\Lambda) \frac{\Delta, X; \Gamma \vdash t_1 : G_1 \rightsquigarrow t'_1 \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \Lambda X.t_1 : \forall X.G_1 \rightsquigarrow \text{reflEv}(\forall X.G_1) (\Lambda X.t'_1) :: \forall X.G_1}$$

Since $\text{reflEv}(\forall X.G_1) : \forall X.G_1 \sim \forall X.G_1$, by (GΛ) and (Gasc) rules, we get that

$$\Delta; \Gamma \vdash \text{reflEv}(\forall X.G_1) (\Lambda X.t'_1) :: \forall X.G_1 : \forall X.G_1$$

Thus, the result follows immediately.

Case (Gasc). We know that $\mathbf{t} = \mathbf{t}_1 :: \mathbf{G}$, and

$$(\mathbf{Gasc}) \frac{\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}' \quad \mathbf{G}' \sim \mathbf{G}}{\Delta; \Gamma \vdash \mathbf{t}_1 :: \mathbf{G} : \mathbf{G}}$$

Since $\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}'$, by the induction hypothesis, we get that $\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}' \rightsquigarrow \mathbf{t}'_1$ and $\Delta; \Gamma \vdash \mathbf{t}'_1 : \mathbf{G}'$. Since $\mathbf{G}' \sim \mathbf{G}$, by Lemma B.104, we get that $\text{initEv}(\mathbf{G}', \mathbf{G}) : \mathbf{G}' \sim \mathbf{G}$. Thus, we get that

$$(\mathbf{EGasc}) \frac{\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}' \rightsquigarrow \mathbf{t}'_1 \quad \varepsilon = \text{initEv}(\mathbf{G}', \mathbf{G})}{\Delta; \Gamma \vdash \mathbf{t}_1 :: \mathbf{G} : \mathbf{G} \rightsquigarrow \varepsilon \mathbf{t}'_1 :: \mathbf{G}}$$

Therefore, by (Gasc) rule, we get that $\Delta; \Gamma \vdash \varepsilon \mathbf{t}'_1 :: \mathbf{G} : \mathbf{G}$, and the result follows immediately.

Case (Gpairi). We know that $\mathbf{t} = \pi_i(\mathbf{t}_1)$, $\mathbf{G} = \text{proj}_i^\sharp(\mathbf{G}'_1)$ and

$$(\mathbf{Gpairi}) \frac{\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}'_1}{\Delta; \Gamma \vdash \pi_i(\mathbf{t}_1) : \text{proj}_i^\sharp(\mathbf{G}'_1)}$$

Since $\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}'_1$, by the induction hypothesis, we get that $\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}'_1 \rightsquigarrow \mathbf{t}'_1$ and $\Delta; \Gamma \vdash \mathbf{t}'_1 : \mathbf{G}'_1$. Since $\text{proj}_i^\sharp(\mathbf{G}'_1)$ is defined, by Lemma B.137, we know that $\mathbf{G}'_1 \rightarrow \mathbf{G}_1 \times \mathbf{G}_2$ is defined and $\text{proj}_i^\sharp(\mathbf{G}'_1) = \mathbf{G}_i$, for some \mathbf{G}_1 and \mathbf{G}_2 . Since $\mathbf{G}'_1 \rightarrow \mathbf{G}_1 \times \mathbf{G}_2$, by Lemmas B.129 and B.104, we know that $\varepsilon = \text{initEv}(\mathbf{G}'_1, \mathbf{G}_1 \times \mathbf{G}_2)$ is defined. Thus, we get that

$$(\mathbf{EGpairi}) \frac{\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}'_1 \rightsquigarrow \mathbf{t}'_1 \quad \mathbf{G}'_1 \rightarrow \mathbf{G}_1 \times \mathbf{G}_2 \quad \varepsilon = \text{initEv}(\mathbf{G}'_1, \mathbf{G}_1 \times \mathbf{G}_2)}{\Delta; \Gamma \vdash \pi_i(\mathbf{t}_1) : \mathbf{G}_i \rightsquigarrow \pi_i(\varepsilon \mathbf{t}'_1 :: \mathbf{G}_1 \times \mathbf{G}_2)}$$

Therefore, by (Gpairi) and (Gasc) rules, we get that $\Delta; \Gamma \vdash \pi_i(\varepsilon \mathbf{t}'_1 :: \mathbf{G}_1 \times \mathbf{G}_2) : \mathbf{G}_i$, and the result follows immediately.

Case (GappG). We know that $\mathbf{t} = \mathbf{t}_1 [\mathbf{F}]$, $\mathbf{G} = \text{inst}^\sharp(\mathbf{G}'_1, \mathbf{F})$ and

$$(\mathbf{GappG}) \frac{\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}'_1 \quad \Delta \vdash \mathbf{F}}{\Delta; \Gamma \vdash \mathbf{t}_1 [\mathbf{F}] : \text{inst}^\sharp(\mathbf{G}'_1, \mathbf{F})}$$

Since $\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}'_1$, by the induction hypothesis, we get that $\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}'_1 \rightsquigarrow \mathbf{t}'_1$ and $\Delta; \Gamma \vdash \mathbf{t}'_1 : \mathbf{G}'_1$. Since $\text{inst}^\sharp(\mathbf{G}'_1, \mathbf{F})$ is defined, by Lemma B.138, we know that $\mathbf{G}'_1 \rightarrow \forall \mathbf{X}. \mathbf{G}_1$ is defined and $\text{inst}^\sharp(\mathbf{G}'_1, \mathbf{F}) = \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}$, for some \mathbf{G}_1 . Since $\mathbf{G}'_1 \rightarrow \forall \mathbf{X}. \mathbf{G}_1$, by Lemmas B.129 and B.104, we know that $\varepsilon = \text{initEv}(\mathbf{G}'_1, \forall \mathbf{X}. \mathbf{G}_1)$ is defined. Thus, we get that

$$(\mathbf{EGappG}) \frac{\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}'_1 \rightsquigarrow \mathbf{t}'_1 \quad \Delta \vdash \mathbf{F} \quad \mathbf{G}'_1 \rightarrow \forall \mathbf{X}. \mathbf{G}_1}{\varepsilon_1 = \text{initEv}(\mathbf{G}'_1, \forall \mathbf{X}. \mathbf{G}_1) \quad \varepsilon_2 = \text{instEv}(\mathbf{G}_1, \mathbf{X}, \mathbf{F})} \frac{}{\Delta; \Gamma \vdash \mathbf{t}_1 [\mathbf{F}] : \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X} \rightsquigarrow \varepsilon_2((\varepsilon_1 \mathbf{t}'_1 :: \forall \mathbf{X}. \mathbf{G}_1) [\mathbf{F}]) :: \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}}$$

Therefore, by (Gasc) and (GappG) rules, we get that $\Delta; \Gamma \vdash \varepsilon_2((\varepsilon_1 \mathbf{t}'_1 :: \forall \mathbf{X}. \mathbf{G}_1) [\mathbf{F}]) :: \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X} : \mathbf{G}_1[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}$, and the result follows immediately.

Case (Gapp). We know that $\mathbf{t} = \mathbf{t}_1 \mathbf{t}_2$, $\mathbf{G} = \text{cod}^\sharp(\mathbf{G}_1)$ and

$$(\mathbf{Gapp}) \frac{\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}_1 \quad \Delta; \Gamma \vdash \mathbf{t}_2 : \mathbf{G}_2}{\text{dom}^\sharp(\mathbf{G}_1) \sim \mathbf{G}_2} \frac{}{\Delta; \Gamma \vdash \mathbf{t}_1 \mathbf{t}_2 : \text{cod}^\sharp(\mathbf{G}_1)}$$

Since $\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}_1$ and $\Delta; \Gamma \vdash \mathbf{t}_2 : \mathbf{G}_2$, by the induction hypothesis, we get that $\Delta; \Gamma \vdash \mathbf{t}_1 : \mathbf{G}_1 \rightsquigarrow \mathbf{t}'_1$, $\Delta; \Gamma \vdash \mathbf{t}'_1 : \mathbf{G}_1$, $\Delta; \Gamma \vdash \mathbf{t}_2 : \mathbf{G}_2 \rightsquigarrow \mathbf{t}'_2$ and $\Delta; \Gamma \vdash \mathbf{t}'_2 : \mathbf{G}_2$. Since $\text{cod}^\sharp(\mathbf{G}_1)$ is defined, by Lemma B.136, we know that $\mathbf{G}_1 \rightarrow \mathbf{G}_{11} \rightarrow \mathbf{G}_{12}$ is defined and $\text{dom}(\mathbf{G}_1) = \mathbf{G}_{11}$ and $\text{cod}(\mathbf{G}_1) = \mathbf{G}_{12}$, for some \mathbf{G}_{11} and

\mathbf{G}_{12} . Since $\mathbf{G}_1 \rightarrow \mathbf{G}_{11} \rightarrow \mathbf{G}_{12}$, by Lemmas B.129 and B.104, we know that $\varepsilon_1 = \text{initEv}(\mathbf{G}_1, \mathbf{G}_{11} \rightarrow \mathbf{G}_{12})$ is defined. Since $\text{dom}^\sharp(\mathbf{G}_1) \sim \mathbf{G}_2$, by Lemma B.104, we know that $\varepsilon_2 = \text{initEv}(\mathbf{G}_2, \mathbf{G}_{11})$ is defined. Thus, we get that

$$(\mathbf{E}_{\text{Gapp}}) \frac{\Delta; \Gamma \vdash t_1 : \mathbf{G}_1 \rightsquigarrow t'_1 \quad \Delta; \Gamma \vdash t_2 : \mathbf{G}_2 \rightsquigarrow t'_2 \quad \mathbf{G}_1 \rightarrow \mathbf{G}_{11} \rightarrow \mathbf{G}_{12} \quad \varepsilon_1 = \text{initEv}(\mathbf{G}_1, \mathbf{G}_{11} \rightarrow \mathbf{G}_{12}) \quad \varepsilon_2 = \text{initEv}(\mathbf{G}_2, \mathbf{G}_{11})}{\Delta; \Gamma \vdash t_1 t_2 : \mathbf{G}_{12} \rightsquigarrow (\varepsilon_1 t'_1 :: \mathbf{G}_{11} \rightarrow \mathbf{G}_{12}) (\varepsilon_2 t'_2 :: \mathbf{G}_{11})}$$

Therefore, by $(\mathbf{G}_{\text{asc}})$ and $(\mathbf{G}_{\text{app}})$ rules, we get that $\Delta; \Gamma \vdash (\varepsilon_1 t'_1 :: \mathbf{G}_{11} \rightarrow \mathbf{G}_{12}) (\varepsilon_2 t'_2 :: \mathbf{G}_{11}) : \mathbf{G}_{12}$, and the result follows immediately. \square

B.6.4 Gradual Guarantees

Definition B.108

$$\frac{}{\cdot \sqsubseteq \cdot} \quad \frac{\Gamma_1 \sqsubseteq \Gamma_2 \quad \mathbf{G}_1 \sqsubseteq \mathbf{G}_2}{\Gamma_1, x : \mathbf{G}_1 \sqsubseteq \Gamma_2, x : \mathbf{G}_2}$$

Definition B.109

$$\frac{}{(\cdot, \cdot) \equiv \cdot} \quad \frac{(\Gamma_1, \Gamma_2) \equiv \Omega \quad \mathbf{G}_1 \sqsubseteq \mathbf{G}_2}{(\Gamma_1, x : \mathbf{G}_1, \Gamma_2, x : \mathbf{G}_2) \equiv \Omega, x : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2}$$

Definition B.110

$$\frac{}{\cdot \rightsquigarrow \cdot} \quad \frac{\Omega \rightsquigarrow \Omega \quad \mathbf{c} = \text{initPT}(\mathbf{G}_1, \mathbf{G}_2)}{\Omega, x \mapsto \mathbf{G}_1 \sqsubseteq \mathbf{G}_2 \rightsquigarrow \Omega, x \mapsto \mathbf{c} : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2}$$

Lemma B.111 *If $\Omega \vdash t_1 : \mathbf{G}_1 \sqsubseteq t_2 : \mathbf{G}_2$, $\Delta; \Gamma_1 \vdash t_1 : \mathbf{G}_1$ and $(\Gamma_1, \Gamma_2) \equiv \Omega$, then $\Delta; \Gamma_2 \vdash t_2 : \mathbf{G}_2$, $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, $\Gamma_1 \sqsubseteq \Gamma_2$.*

PROOF. The proof proceed by induction on the typing derivation $\Delta; \Gamma_1 \vdash t_1 : \mathbf{G}_1$. By Lemma B.126, we get that $\Gamma_1 \sqsubseteq \Gamma_2$.

Case (\mathbf{G}_b) . Trivial by definition of term precision using (\sqsubseteq_b) .

$$\sqsubseteq_b \frac{}{\Omega \vdash b : \mathbf{B} \sqsubseteq b : \mathbf{B}}$$

Note that $\mathbf{B} \sqsubseteq \mathbf{B}$.

Case (\mathbf{G}_x) . Trivial by definition of term precision using (\sqsubseteq_x) .

$$\sqsubseteq_x \frac{x : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2 \in \Omega}{\Omega \vdash x : \mathbf{G}_1 \sqsubseteq x : \mathbf{G}_2}$$

Note that since $x : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2 \in \Omega$, by Definition B.109, we know that $x : \mathbf{G}_2 \in \Gamma_2$ and $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$.

Case (\mathbf{G}_λ) . Then $t_1 = (\lambda x : \mathbf{G}'_1. t)$ and $\mathbf{G}_1 = \mathbf{G}'_1 \rightarrow \mathbf{G}'_2$. By (\mathbf{G}_λ) we know that:

$$(\mathbf{G}_\lambda) \frac{\Delta; \Gamma_1, x : \mathbf{G}'_1 \vdash t : \mathbf{G}'_2}{\Delta; \Gamma_1 \vdash \lambda x : \mathbf{G}'_1. t : \mathbf{G}'_1 \rightarrow \mathbf{G}'_2} \tag{B.1}$$

Consider t_2 such that $\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2$. By definition of term precision, t_2 must have the form $t_2 = (\lambda x : G'_1.t')$ and therefore

$$(\sqsubseteq_\lambda) \frac{\Omega, x : G'_1 \sqsubseteq G''_1 \vdash t : G'_2 \sqsubseteq t' : G''_2 \quad G'_1 \sqsubseteq G''_1}{\Omega \vdash (\lambda x : G'_1.t) : G'_1 \rightarrow G'_2 \sqsubseteq (\lambda x : G''_1.t') : G''_1 \rightarrow G''_2} \quad (\text{B.2})$$

Using induction hypotheses on $\Delta; \Gamma_1, x : G'_1 \vdash t : G'_2$, we get that $\Delta; \Gamma_2, x : G''_1 \vdash t' : G''_2$ and $G'_2 \sqsubseteq G''_2$. Then we can use rule (G λ) to derive:

$$(\text{G}\lambda) \frac{\Delta; \Gamma_2, x : G''_1 \vdash t' : G''_2}{\Delta; \Gamma_2 \vdash (\lambda x : G''_1.t') : G''_1 \rightarrow G''_2}$$

Since $G'_1 \sqsubseteq G''_1$ and $G'_2 \sqsubseteq G''_2$, we get that $G'_1 \rightarrow G'_2 \sqsubseteq G''_1 \rightarrow G''_2$, by Definition of Type Precision, and the result holds.

Case (G Λ). Then $t_1 = (\Lambda X.t)$ and $G_1 = \forall X.G'_1$. By (G Λ) we know that:

$$(\text{G}\Lambda) \frac{\Delta, X; \Gamma_1 \vdash t : G'_1}{\Delta; \Gamma_1 \vdash (\Lambda X.t) : \forall X.G'_1} \quad (\text{B.3})$$

Consider t_2 such that $\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2$. By definition of term precision t_2 must have the form $t_2 = (\Lambda X.t')$ and therefore

$$(\sqsubseteq_\Lambda) \frac{\Omega \vdash t : G'_1 \sqsubseteq t' : G''_1}{\Omega \vdash (\Lambda X.t) : \forall X.G'_1 \sqsubseteq (\Lambda X.t') : \forall X.G''_1} \quad (\text{B.4})$$

Using the induction hypotheses on $\Delta, X; \Gamma_1 \vdash t : G'_1$, we get that $\Delta, X; \Gamma_2 \vdash t' : G''_1$ and $G'_1 \sqsubseteq G''_1$. Then we can use rule (G Λ) to derive:

$$(\text{G}\Lambda) \frac{\Delta, X; \Gamma_2 \vdash t' : G''_1}{\Delta; \Gamma_2 \vdash (\Lambda X.t') : \forall X.G''_1}$$

Since $G'_1 \sqsubseteq G''_1$, we get that $\forall X.G'_1 \sqsubseteq \forall X.G''_1$, by Definition of Type Precision, and the result holds.

Case (Gpair). Then $t_1 = \langle t'_1, t'_2 \rangle$ and $G_1 = G'_1 \times G'_2$. By (Gpair) we know that:

$$(\text{Gpair}) \frac{\Delta; \Gamma_1 \vdash t'_1 : G'_1 \quad \Delta; \Gamma_1 \vdash t'_2 : G'_2}{\Delta; \Gamma_1 \vdash \langle t'_1, t'_2 \rangle : G'_1 \times G'_2} \quad (\text{B.5})$$

Consider t_2 such that $\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2$. By definition of term precision, t_2 must have the form $\langle t''_1, t''_2 \rangle$ and therefore

$$(\sqsubseteq_\times) \frac{\Omega \vdash t'_1 : G'_1 \sqsubseteq t''_1 : G''_1 \quad \Omega \vdash t'_2 : G'_2 \sqsubseteq t''_2 : G''_2}{\Omega \vdash \langle t'_1, t'_2 \rangle : G'_1 \times G'_2 \sqsubseteq \langle t''_1, t''_2 \rangle : G''_1 \times G''_2} \quad (\text{B.6})$$

Using induction hypotheses on $\Delta; \Gamma_1 \vdash t'_1 : G'_1$ and $\Delta; \Gamma_1 \vdash t'_2 : G'_2$, we get that $\Delta; \Gamma_2 \vdash t''_1 : G''_1$, $\Delta; \Gamma_2 \vdash t''_2 : G''_2$, $G'_1 \sqsubseteq G''_1$ and $G'_2 \sqsubseteq G''_2$. Then we can use rule (Gpair) to derive:

$$(\text{Gpair}) \frac{\Delta; \Gamma_2 \vdash t''_1 : G''_1 \quad \Delta; \Gamma_2 \vdash t''_2 : G''_2}{\Delta; \Gamma_2 \vdash \langle t''_1, t''_2 \rangle : G''_1 \times G''_2}$$

Since $G'_1 \sqsubseteq G''_1$ and $G'_2 \sqsubseteq G''_2$, we get that $G'_1 \times G'_2 \sqsubseteq G''_1 \times G''_2$, by Definition of Type Precision, and the result holds.

Case (Gasc). Then $t_1 = t :: G_1$. By (Gasc) we know that:

$$(Gasc) \frac{\Delta; \Gamma_1 \vdash t : G \quad G \sim G_1}{\Delta; \Gamma_1 \vdash t :: G_1 : G_1} \quad (B.7)$$

Consider t_2 such that $\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2$. By definition of term precision t_2 must have the form $t_2 = t' :: G_2$ and therefore

$$(\sqsubseteq asc) \frac{\Omega \vdash t : G \sqsubseteq t' : G' \quad G_1 \sqsubseteq G_2}{\Omega \vdash t :: G_1 : G_1 \sqsubseteq t' :: G_2 : G_2} \quad (B.8)$$

Using induction hypotheses on $\Delta; \Gamma_1 \vdash t : G$, we get that $\Delta; \Gamma_2 \vdash t' : G'$ and $G \sqsubseteq G'$. $\Delta; \Gamma \vdash t' : G'$, where $G \sqsubseteq G'$. We can use rule (Gasc) and Lemma B.130 to derive:

$$(Gasc) \frac{\Delta; \Gamma_2 \vdash t' : G' \quad G' \sim G_2}{\Delta; \Gamma_2 \vdash t' :: G_2 : G_2}$$

By premise, we know that $G_1 \sqsubseteq G_2$. Therefore, the result holds.

Case (Gapp). Then $t_1 = t'_1 t'_2$ and $G_1 = \text{cod}^\sharp(G'_1)$. By (Gapp) we know that:

$$(Gapp) \frac{\Delta; \Gamma_1 \vdash t'_1 : G'_1 \quad \Delta; \Gamma_1 \vdash t'_2 : G'_2 \quad \text{dom}^\sharp(G'_1) \sim G'_2}{\Delta; \Gamma_1 \vdash t'_1 t'_2 : \text{cod}^\sharp(G'_1)} \quad (B.9)$$

Consider t_2 such that $\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2$. By definition of term precision t_2 must have the form $t_2 = t''_1 t''_2$ and therefore

$$\sqsubseteq_{app} \frac{\Omega \vdash t'_1 : G'_1 \sqsubseteq t''_1 : G''_1 \quad \Omega \vdash t'_2 : G'_2 \sqsubseteq t''_2 : G''_2 \quad G'_1 \rightarrow G_{11} \rightarrow G_{12} \quad G''_1 \rightarrow G_{21} \rightarrow G_{22}}{\Omega \vdash t'_1 t'_2 : G_{12} \sqsubseteq t''_1 t''_2 : G_{22}} \quad (B.10)$$

Using the induction hypotheses on $\Delta; \Gamma_1 \vdash t'_1 : G'_1$ and $\Delta; \Gamma_1 \vdash t'_2 : G'_2$, we get that $\Delta; \Gamma_2 \vdash t''_1 : G''_1$, $\Delta; \Gamma_2 \vdash t''_2 : G''_2$, $G'_1 \sqsubseteq G''_1$ and $G'_2 \sqsubseteq G''_2$. By definition type precision and the definition of dom^\sharp , we know that $\text{dom}^\sharp(G'_1) \sqsubseteq \text{dom}^\sharp(G''_1)$ and, therefore by Lemma B.130, we get that $\text{dom}^\sharp(G''_1) \sim G''_2$. Also, by the previous argument, we get that $\text{cod}^\sharp(G'_1) \sqsubseteq \text{cod}^\sharp(G''_1)$. Note that since $G''_1 \rightarrow G_{21} \rightarrow G_{22}$, we know that $\text{dom}^\sharp(G''_1) = G_{21}$ and $\text{cod}^\sharp(G''_1) = G_{22}$, by Lemma B.136. Then we can use rule (Gapp) to derive:

$$(Gapp) \frac{\Delta; \Gamma_2 \vdash t''_1 : G''_1 \quad \Delta; \Gamma_2 \vdash t''_2 : G''_2 \quad \text{dom}^\sharp(G''_1) \sim G''_2}{\Delta; \Gamma_2 \vdash t''_1 t''_2 : \text{cod}^\sharp(G''_1)}$$

and the result holds.

Case (GappG). Then $t_1 = t [F]$. By (GappG) we know that:

$$(GappG) \frac{\Delta; \Gamma_1 \vdash t : G'_1 \quad \Delta \vdash F}{\Delta; \Gamma_1 \vdash t [F] : \text{inst}^\sharp(G'_1, F)} \quad (B.11)$$

where $G_1 = \text{inst}^\sharp(G'_1, F)$. Consider t_2 such that $\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2$. By definition of term precision t_2 must have the form $t_2 = t' [F]$ and therefore

$$(\sqsubseteq_{appG}) \frac{\Omega \vdash t : G'_1 \sqsubseteq t' : G'_2 \quad G'_1 \leq G'_2 \quad G'_1 \rightarrow \forall X. G''_1 \quad G'_2 \rightarrow \forall X. G''_2}{\Omega \vdash t [F] : G''_1[F/X] \sqsubseteq t' [F] : G''_2[F/X]} \quad (B.12)$$

Using induction hypotheses on $\Delta; \Gamma_1 \vdash t : G'_1$, we get that $\Delta; \Gamma_2 \vdash t' : G'_2$ and $G'_1 \sqsubseteq G'_2$. We can use rule (GappG) to derive:

$$\text{(Gasc)} \frac{\Delta; \Gamma_2 \vdash t' : G'_2 \quad \Delta \vdash F}{\Delta; \Gamma_2 \vdash t' [F] : \text{inst}^\sharp(G'_2, F)}$$

Finally, by Lemma B.141 we know that $\text{inst}^\sharp(G'_1, F) \sqsubseteq \text{inst}^\sharp(G'_2, F)$ and since $G'_2 \rightarrow \forall X.G''_2$, by Lemma B.138, we know that $\text{inst}^\sharp(G'_2, F) = G''_2[F/X]\backslash X$. Thus, the result holds.

Case (Gpairi). Then $t_1 = \pi_i(t)$ and $G_1 = \text{proj}_i^\sharp(G)$. By (Gpair) we know that:

$$\text{(Gpairi)} \frac{\Delta; \Gamma_1 \vdash t : G}{\Delta; \Gamma_1 \vdash \pi_i(t) : \text{proj}_i^\sharp(G)} \quad (\text{B.13})$$

Consider t_2 such that $\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2$. By definition of term precision, t_2 must have the form $\pi_i(t')$ and therefore

$$\text{(\sqsubseteqpairi)} \frac{\Omega \vdash t : G \sqsubseteq t' : G' \quad G \rightarrow G_{11} \times G_{12} \quad G' \rightarrow G_{21} \times G_{22}}{\Omega \vdash \pi_i(t) : G_{1i} \sqsubseteq \pi_i(t') : G_{2i}} \quad (\text{B.14})$$

Using induction hypotheses on the premises $\Delta; \Gamma_1 \vdash t : G$, we get that $\Delta; \Gamma_2 \vdash t' : G'$ and $G \sqsubseteq G'$. Then we can use rule (Gpairi) to derive:

$$\text{(Gpairi)} \frac{\Delta; \Gamma_2 \vdash t' : G'}{\Delta; \Gamma_2 \vdash \pi_i(t') : \text{proj}_i^\sharp(G')}$$

Finally, by Lemma B.140 we can infer that $\text{proj}_i^\sharp(G) \sqsubseteq \text{proj}_i^\sharp(G')$ and since $G' \rightarrow G_{21} \times G_{22}$, by Lemma B.137, we know that $\text{proj}_i^\sharp(G') = G_{2i}$. Thus, the result holds. □

Lemma B.112 *If $c : G \sqsubseteq G'$ then $c = \text{initPT}(G, G')$.*

PROOF. We proceed by induction on $c : G \sqsubseteq G'$.

Case (B : B \sqsubseteq B). Since, $\text{initPT}(B, B) = B$, the result holds.

Case (X : X \sqsubseteq X). Since, $\text{initPT}(X, X) = X$, the result holds.

Case ($c_1 \rightarrow c_2 : G_{11} \rightarrow G_{12} \sqsubseteq G_{21} \rightarrow G_{22}$). We know that $c_1 : G_{11} \sqsubseteq G_{21}$ and $c_2 : G_{12} \sqsubseteq G_{22}$. By the inductive hypothesis, we get that $c_1 = \text{initPT}(G_{11}, G_{21})$ and $c_2 = \text{initPT}(G_{12}, G_{22})$. We are required to prove that $c = c_1 \rightarrow c_2 = \text{initPT}(G_{11} \rightarrow G_{12}, G_{21} \rightarrow G_{22}) = \text{initPT}(G_{11}, G_{21}) \rightarrow \text{initPT}(G_{12}, G_{22})$, as we already showed.

Case ($c_1 \times c_2 : G_{11} \times G_{12} \sqsubseteq G_{21} \times G_{22}$). We know that $c_1 : G_{11} \sqsubseteq G_{21}$ and $c_2 : G_{12} \sqsubseteq G_{22}$. By the inductive hypothesis, we get that $c_1 = \text{initPT}(G_{11}, G_{21})$ and $c_2 = \text{initPT}(G_{12}, G_{22})$. We are required to prove that $c = c_1 \times c_2 = \text{initPT}(G_{11} \times G_{12}, G_{21} \times G_{22}) = \text{initPT}(G_{11}, G_{21}) \times \text{initPT}(G_{12}, G_{22})$, as we already showed.

Case ($\forall X.c_1 : \forall X.G_{11} \sqsubseteq \forall X.G_{21}$). We know that $c_1 : G_{11} \sqsubseteq G_{21}$. By the inductive hypothesis, we get that $c_1 = \text{initPT}(G_{11}, G_{21})$. We are required to prove that $c = \forall X.c_1 = \text{initPT}(\forall X.G_{11}, \forall X.G_{21}) = \forall X.\text{initPT}(G_{11}, G_{21})$, as we already showed.

Case ($\forall X.c_{21} : \forall X.G_{21} \sqsubseteq \forall X.G_{31}$). Since $G_2 = \forall X.G_{21}$, by Lemma B.11, we know that $c_1 = \forall X.c_{11}$ and $G_1 = \forall X.G_{11}$. We know that $c_{21} : G_{21} \sqsubseteq G_{31}$ and $c_{11} : G_{11} \sqsubseteq G_{21}$. By the inductive hypothesis, we get that $c_{11}; c_{21} = c_{31}$. Therefore, we get that $\forall X.c_{11}; \forall X.c_{21} = \forall X.c_{31}$, and the result holds.

Case ($\text{inj}_B : B \sqsubseteq ?_\delta$). Since, $\text{initPT}(B, ?_\delta) = \text{inj}_B$, the result holds.

Case ($\text{inj}_X : F \sqsubseteq ?_\delta \wedge X : F \in \delta$). We have the following cases.

- $F = B$. This case is not possible because $X : B \notin \delta$, we have that $X : X\delta$.
- $F = Y$. We know that $X : Y \in \delta$, therefore $X = Y$. We know that $\text{initPT}(X, ?_\delta) = \text{inj}_X$, and the result holds.

Case ($\text{inj}_\rightarrow(\mathbf{c}') : G \sqsubseteq ?_\delta$). We know that $\mathbf{c}' : G \sqsubseteq ?_\delta \rightarrow ?_\delta$ and $G = G_{11} \rightarrow G_{21}$. By the inductive hypothesis, we get that $\mathbf{c}' = \text{initPT}(G, ?_\delta \rightarrow ?_\delta)$. We are required to prove that $\mathbf{c} = \text{inj}_\rightarrow(\mathbf{c}') = \text{initPT}(G, ?_\delta) = \text{inj}_\rightarrow(\text{initPT}(G, ?_\delta \rightarrow ?_\delta))$, as we already showed.

Case ($\text{inj}_\times(\mathbf{c}') : G \sqsubseteq ?_\delta$). We know that $\mathbf{c}' : G \sqsubseteq ?_\delta \times ?_\delta$ and $G = G_{11} \times G_{21}$. By the inductive hypothesis, we get that $\mathbf{c}' = \text{initPT}(G, ?_\delta \times ?_\delta)$. We are required to prove that $\mathbf{c} = \text{inj}_\times(\mathbf{c}') = \text{initPT}(G, ?_\delta) = \text{inj}_\times(\text{initPT}(G, ?_\delta \times ?_\delta))$, as we already showed.

Case ($\text{inj}_\forall(\mathbf{c}') : G \sqsubseteq ?_\delta$). We know that $\mathbf{c}' : G \sqsubseteq \forall X. ?_\delta, X : X$ and $G = \forall X. G_{11}$. By the inductive hypothesis, we get that $\mathbf{c}' = \text{initPT}(G, \forall X. ?_\delta, X : X)$. We are required to prove that $\mathbf{c} = \text{inj}_\forall(\mathbf{c}') = \text{initPT}(G, ?_\delta) = \text{inj}_\forall(\text{initPT}(G, \forall X. ?_\delta, X : X))$, as we already showed.

Case ($\text{inj}_? : ?_\delta \sqsubseteq ?_{\delta'}$). The result holds immediately since $\text{initPT}(?_\delta, ?_{\delta'})$, as we already know.

□

Lemma B.113 *If $\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2$ then $G_1 \sqsubseteq G_2$.*

PROOF. Straightforward induction on $\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2$.

□

Lemma B.114 $\text{initPT}(G, G) = \text{refl}_G^{\sqsubseteq}$.

PROOF. We proceed by induction on G .

Case ($G = B$). Then, we know that $\text{initPT}(B, B) = \text{refl}_B^{\sqsubseteq} = B$. Therefore, the result follows immediately.

Case ($G = X$). Then, we know that $\text{initPT}(X, X) = \text{refl}_X^{\sqsubseteq} = X$. Therefore, the result follows immediately.

Case ($G = G_1 \rightarrow G_2$). Then, we know that

- $\text{initPT}(G_1 \rightarrow G_2, G_1 \rightarrow G_2) = \text{initPT}(G_1, G_1) \rightarrow \text{initPT}(G_2, G_2)$
- $\text{refl}_{G_1 \rightarrow G_2}^{\sqsubseteq} = \text{refl}_{G_1}^{\sqsubseteq} \rightarrow \text{refl}_{G_2}^{\sqsubseteq}$

By the induction hypothesis on G_1 and G_2 , we get that $\text{initPT}(G_1, G_1) = \text{refl}_{G_1}^{\sqsubseteq}$ and $\text{initPT}(G_2, G_2) = \text{refl}_{G_2}^{\sqsubseteq}$. Therefore, $\text{initPT}(G_1, G_1) \rightarrow \text{initPT}(G_2, G_2) = \text{refl}_{G_1}^{\sqsubseteq} \rightarrow \text{refl}_{G_2}^{\sqsubseteq}$, and the result follows immediately.

Case ($G = G_1 \times G_2$). Then, we know that

- $\text{initPT}(G_1 \times G_2, G_1 \times G_2) = \text{initPT}(G_1, G_1) \times \text{initPT}(G_2, G_2)$
- $\text{refl}_{G_1 \times G_2}^{\sqsubseteq} = \text{refl}_{G_1}^{\sqsubseteq} \times \text{refl}_{G_2}^{\sqsubseteq}$

By the induction hypothesis on G_1 and G_2 , we get that $\text{initPT}(G_1, G_1) = \text{refl}_{G_1}^{\sqsubseteq}$ and $\text{initPT}(G_2, G_2) = \text{refl}_{G_2}^{\sqsubseteq}$. Therefore, $\text{initPT}(G_1, G_1) \times \text{initPT}(G_2, G_2) = \text{refl}_{G_1}^{\sqsubseteq} \times \text{refl}_{G_2}^{\sqsubseteq}$, and the result follows immediately.

Case ($G = \forall X.G_1$). Then, we know that

- $\text{initPT}(\forall X.G_1, \forall X.G_1) = \forall X.\text{initPT}(G_1, G_1)$
- $\text{refl}_{\forall X.G_1}^{\sqsubseteq} = \forall X.\text{refl}_{G_1}^{\sqsubseteq}$

By the induction hypothesis on G_1 , we get that $\text{initPT}(G_1, G_1) = \text{refl}_{G_1}^{\sqsubseteq}$. Therefore, $\forall X.\text{initPT}(G_1, G_1) = \forall X.\text{refl}_{G_1}^{\sqsubseteq}$, and the result follows immediately.

Case ($G = ?_\delta$). Then, we know that $\text{initPT}(?_\delta, ?_\delta) = \text{refl}_{?_\delta}^{\sqsubseteq} = \text{inj}_?$. Therefore, the result follows immediately. □

Lemma B.115 *If* $G \sqsubseteq ?_\delta$ *then* $\text{initPT}(G, ?_\delta) = \text{inject}_G^{\sqsubseteq}$.

PROOF. We proceed by induction on G . Note that since $G \sqsubseteq ?_\delta$, then $\text{initPT}(G, ?_\delta)$ is defined. By Lemma B.99, we get that $\text{ftv}(G) \subseteq \delta$. Thus, $\text{inject}_G^{\sqsubseteq}$ is defined.

Case ($G = B$). Then, we know that $\text{initPT}(B, ?_\delta) = \text{inject}_B^{\sqsubseteq} = \text{inj}_B$. Therefore, the result follows immediately.

Case ($G = X$). Then, we know that $\text{initPT}(X, ?_\delta) = \text{inject}_X^{\sqsubseteq} = \text{inj}_X$. Therefore, the result follows immediately.

Case ($G = G_1 \rightarrow G_2$). Then, we know that

- $\text{initPT}(G_1 \rightarrow G_2, ?_\delta) = \text{inj}_{\rightarrow}(\text{initPT}(G_1, ?_\delta) \rightarrow \text{initPT}(G_2, ?_\delta))$
- $\text{inject}_{G_1 \rightarrow G_2}^{\sqsubseteq} = \text{inj}_{\rightarrow}(\text{inject}_{G_1}^{\sqsubseteq} \rightarrow \text{inject}_{G_2}^{\sqsubseteq})$

By the induction hypothesis on G_1 and G_2 , we get that $\text{initPT}(G_1, ?_\delta) = \text{inject}_{G_1}^{\sqsubseteq}$ and $\text{initPT}(G_2, ?_\delta) = \text{inject}_{G_2}^{\sqsubseteq}$. Therefore, $\text{inj}_{\rightarrow}(\text{initPT}(G_1, ?_\delta) \rightarrow \text{initPT}(G_2, ?_\delta)) = \text{inj}_{\rightarrow}(\text{inject}_{G_1}^{\sqsubseteq} \rightarrow \text{inject}_{G_2}^{\sqsubseteq})$, and the result follows immediately.

Case ($G = G_1 \times G_2$). Then, we know that

- $\text{initPT}(G_1 \times G_2, ?_\delta) = \text{inj}_{\times}(\text{initPT}(G_1, ?_\delta) \times \text{initPT}(G_2, ?_\delta))$
- $\text{inject}_{G_1 \times G_2}^{\sqsubseteq} = \text{inj}_{\times}(\text{inject}_{G_1}^{\sqsubseteq} \times \text{inject}_{G_2}^{\sqsubseteq})$

By the induction hypothesis on G_1 and G_2 , we get that $\text{initPT}(G_1, ?_\delta) = \text{inject}_{G_1}^{\sqsubseteq}$ and $\text{initPT}(G_2, ?_\delta) = \text{inject}_{G_2}^{\sqsubseteq}$. Therefore, $\text{inj}_{\times}(\text{initPT}(G_1, ?_\delta) \times \text{initPT}(G_2, ?_\delta)) = \text{inj}_{\times}(\text{inject}_{G_1}^{\sqsubseteq} \times \text{inject}_{G_2}^{\sqsubseteq})$, and the result follows immediately.

Case ($G = \forall X.G_1$). Then, we know that

- $\text{initPT}(\forall X.G_1, ?_\delta) = \forall X.\text{initPT}(G_1, ?_{\delta, X:X})$
- $\text{inject}_{\forall X.G_1}^{\sqsubseteq} = \forall X.\text{inject}_{G_1}^{\sqsubseteq}$

By the induction hypothesis on G_1 , we get that $\text{initPT}(G_1, ?_{\delta, X:X}) = \text{inject}_{G_1}^{\sqsubseteq}$. Therefore, $\forall X.\text{initPT}(G_1, ?_{\delta, X:X}) = \forall X.\text{inject}_{G_1}^{\sqsubseteq}$, and the result follows immediately.

Case ($G = ?_\delta$). Then, we know that $\text{initPT}(?_\delta, ?_{\delta'}) = \text{inject}_{?_\delta}^{\sqsubseteq} = \text{inj}_?$. Therefore, the result follows immediately. □

Lemma B.116 *If $G \sqsubseteq G'$ then $\text{initPT}(G, G'); \text{inject}_{G'}^{\sqsubseteq} = \text{inject}_G^{\sqsubseteq}$.*

PROOF. We proceed by induction on $G \sqsubseteq G'$.

Case (Base Type). $G = B$ and $G' = B$. We know that $B \sqsubseteq B$. Therefore, we get that $\text{initPT}(G, G') = \text{initPT}(B, B) = B$ and $\text{inject}_G^{\sqsubseteq} = \text{inject}_{G'}^{\sqsubseteq} = \text{inj}_B$. Since $B; \text{inj}_B = \text{inj}_B$ the result follows immediately.

Case (Type Variable). $G = X$ and $G' = X$. We know that $X \sqsubseteq X$. Therefore, we get that $\text{initPT}(G, G') = \text{initPT}(X, X) = X$ and $\text{inject}_G^{\sqsubseteq} = \text{inject}_{G'}^{\sqsubseteq} = \text{inj}_X$. Since $X; \text{inj}_X = \text{inj}_X$ the result follows immediately.

Case (Function Type). $G = G_1 - \longrightarrow G'_1$ and $G' = G_2 - \longrightarrow G'_2$. We know that

$$\frac{G_1 \sqsubseteq G_2 \quad G'_1 \sqsubseteq G'_2}{G_1 - \longrightarrow G'_1 \sqsubseteq G_2 - \longrightarrow G'_2}$$

We are required to prove that $\text{initPT}(G_1 - \longrightarrow G'_1, G_2 - \longrightarrow G'_2); \text{inject}_{G_2 - \longrightarrow G'_2}^{\sqsubseteq} = \text{inject}_{G_1 - \longrightarrow G'_1}^{\sqsubseteq}$.

We know that $\text{initPT}(G_1 - \longrightarrow G'_1, G_2 - \longrightarrow G'_2) = \text{initPT}(G_1, G_2) - \longrightarrow \text{initPT}(G'_1, G'_2)$, $\text{inject}_{G_1 - \longrightarrow G'_1}^{\sqsubseteq} = \text{inject}_{G_1}^{\sqsubseteq} - \longrightarrow \text{inject}_{G'_1}^{\sqsubseteq}$ and $\text{inject}_{G_2 - \longrightarrow G'_2}^{\sqsubseteq} = \text{inject}_{G_2}^{\sqsubseteq} - \longrightarrow \text{inject}_{G'_2}^{\sqsubseteq}$. Thus, we are required to prove that

$$\text{initPT}(G_1, G_2) - \longrightarrow \text{initPT}(G'_1, G'_2); \text{inject}_{G_2}^{\sqsubseteq} - \longrightarrow \text{inject}_{G'_2}^{\sqsubseteq} = \text{inject}_{G_1}^{\sqsubseteq} - \longrightarrow \text{inject}_{G'_1}^{\sqsubseteq}$$

Or what is the same, we are required to prove that $\text{initPT}(G_1, G_2); \text{inject}_{G_2}^{\sqsubseteq} = \text{inject}_{G_1}^{\sqsubseteq}$ and $\text{initPT}(G'_1, G'_2); \text{inject}_{G'_2}^{\sqsubseteq} = \text{inject}_{G'_1}^{\sqsubseteq}$, which follows immediately by the induction hypothesis on $G_1 \sqsubseteq G_2$ and $G'_1 \sqsubseteq G'_2$.

Case (Universal Type). $G = \forall X.G_1$ and $G' = \forall X.G_2$. We know that

$$\frac{G_1 \sqsubseteq G_2}{\forall X.G_1 \sqsubseteq \forall X.G_2}$$

We are required to prove that $\text{initPT}(\forall X.G_1, \forall X.G_2); \text{inject}_{\forall X.G_2}^{\sqsubseteq} = \text{inject}_{\forall X.G_1}^{\sqsubseteq}$.

We know that $\text{initPT}(\forall X.G_1, \forall X.G_2) = \forall X.\text{initPT}(G_1, G_2)$, $\text{inject}_{\forall X.G_1}^{\sqsubseteq} = \forall X.\text{inject}_{G_1}^{\sqsubseteq}$ and $\text{inject}_{\forall X.G_2}^{\sqsubseteq} = \forall X.\text{inject}_{G_2}^{\sqsubseteq}$. Thus, we are required to prove that

$$\forall X.\text{initPT}(G_1, G_2); \forall X.\text{inject}_{G_2}^{\sqsubseteq} = \forall X.\text{inject}_{G_1}^{\sqsubseteq}$$

Or what is the same, we are required to prove that $\text{initPT}(G_1, G_2); \text{inject}_{G_2}^{\sqsubseteq} = \text{inject}_{G_1}^{\sqsubseteq}$. which follows immediately by the induction hypothesis on $G_1 \sqsubseteq G_2$.

Case (Inject Base Type). $G = B$ and $G' = ?_\delta$. We know that $B \sqsubseteq ?_\delta$. Therefore, we get that $\text{initPT}(G, G') = \text{initPT}(B, ?_\delta) = \text{inj}_B$, $\text{inject}_{\overline{G}} = \text{inj}_B$ and $\text{inject}_{\overline{G'}} = \text{inj}_?$. Since $\text{inj}_B; \text{inj}_? = \text{inj}_B$ the result follows immediately.

Case (Inject Type Variable). $G = X$ and $G' = ?_\delta$. We know that

$$\frac{X : F \in \delta}{X \sqsubseteq ?_\delta}$$

Therefore, we get that $\text{initPT}(G, G') = \text{initPT}(X, ?_\delta) = \text{inj}_X$, $\text{inject}_{\overline{G}} = \text{inj}_X$ and $\text{inject}_{\overline{G'}} = \text{inj}_?$. Since $\text{inj}_X; \text{inj}_? = \text{inj}_X$ the result follows immediately.

Case (Inject Function Type). $G = G_1 \rightarrow G_2$ and $G' = ?_\delta$. We know that

$$\frac{G_1 \rightarrow G_2 \sqsubseteq ?_\delta \rightarrow ?_\delta}{G_1 \rightarrow G_2 \sqsubseteq ?_\delta}$$

We are required to prove that $\text{initPT}(G_1 \rightarrow G_2, ?_\delta); \text{inject}_{\overline{?_\delta}} = \text{inject}_{\overline{G_1 \rightarrow G_2}}$.

We know that $\text{initPT}(G_1 \rightarrow G_2, ?_\delta) = \text{inj}_{\rightarrow}(\text{initPT}(G_1, ?_\delta) \rightarrow \text{initPT}(G_2, ?_\delta))$, $\text{inject}_{\overline{G_1 \rightarrow G_2}} = \text{inj}_{\rightarrow}(\text{inject}_{\overline{G_1}} \rightarrow \text{inject}_{\overline{G_2}})$ and $\text{inject}_{\overline{?_\delta}} = \text{inj}_?$. Thus, we are required to prove that

$$\text{inj}_{\rightarrow}(\text{initPT}(G_1, ?_\delta) \rightarrow \text{initPT}(G_2, ?_\delta)); \text{inj}_? = \text{inj}_{\rightarrow}(\text{inject}_{\overline{G_1}} \rightarrow \text{inject}_{\overline{G_2}})$$

Since $\mathbf{c}; \text{inj}_? = \mathbf{c}$, we are required to prove that $\text{initPT}(G_1, ?_\delta) = \text{inject}_{\overline{G_1}}$ and $\text{initPT}(G_2, ?_\delta) = \text{inject}_{\overline{G_2}}$, which follows immediately by Lemma B.115.

Case (Inject Universal Type). $G = \forall X. G_1$ and $G' = ?_\delta$. We know that

$$\frac{\forall X. G_1 \sqsubseteq \forall X. ?_\delta, X : X}{\forall X. G_1 \sqsubseteq ?_\delta}$$

We are required to prove that $\text{initPT}(\forall X. G_1, ?_\delta); \text{inject}_{\overline{?_\delta}} = \text{inject}_{\overline{\forall X. G_1}}$.

We know that $\text{initPT}(\forall X. G_1, ?_\delta) = \text{inj}_{\forall}(\forall X. \text{initPT}(G_1, ?_\delta))$, $\text{inject}_{\overline{\forall X. G_1}} = \text{inj}_{\forall}(\forall X. \text{inject}_{\overline{G_1}})$ and $\text{inject}_{\overline{?_\delta}} = \text{inj}_?$. Thus, we are required to prove that

$$\text{inj}_{\forall}(\forall X. \text{initPT}(G_1, ?_\delta)); \text{inj}_? = \text{inj}_{\forall}(\forall X. \text{inject}_{\overline{G_1}})$$

Since $\mathbf{c}; \text{inj}_? = \mathbf{c}$, we are required to prove that $\text{initPT}(G_1, ?_\delta) = \text{inject}_{\overline{G_1}}$, which follows immediately by Lemma B.115.

Case (Unknown Type). $G = ?_\delta$ and $G = ?_{\delta'}$. We know that

$$\frac{\delta \subseteq \delta'}{?_\delta \sqsubseteq ?_{\delta'}}$$

Therefore, we get that $\text{initPT}(G, G') = \text{initPT}(?_\delta, ?_{\delta'}) = \text{inj}_?$ and $\text{inject}_{\overline{G}} = \text{inject}_{\overline{?_\delta}} = \text{inject}_{\overline{?_{\delta'}}} = \text{inject}_{\overline{?_{\delta'}}} = \text{inj}_?$. Since $\text{inj}_?; \text{inj}_? = \text{inj}_?$, the result follows immediately.

□

Lemma B.117 *If $G_1 \sqsubseteq G_2$, then $G_1 \sqcap G_2 = G_2 \sqcap G_1 = G_1$.*

PROOF. We proceed by induction on $G_1 \sqsubseteq G_2$.

Case ($B \sqsubseteq B$). Then, we get that $G_1 \sqcap G_2 = G_2 \sqcap G_1 = B \sqcap B = B = G_1$. Thus, the result follows immediately.

Case ($X \sqsubseteq X$). Then, we get that $G_1 \sqcap G_2 = G_2 \sqcap G_1 = X \sqcap X = X = G_1$. Thus, the result follows immediately.

Case ($G_{11} \rightarrow G_{12} \sqsubseteq G_{21} \rightarrow G_{22}$). Then, we know that

- $G_{11} \sqsubseteq G_{21}$ and $G_{12} \sqsubseteq G_{22}$
- $G_1 \sqcap G_2 = (G_{11} \rightarrow G_{12}) \sqcap (G_{21} \rightarrow G_{22}) = (G_{11} \sqcap G_{21}) \rightarrow (G_{12} \sqcap G_{22})$
- $G_2 \sqcap G_1 = (G_{21} \rightarrow G_{22}) \sqcap (G_{11} \rightarrow G_{12}) = (G_{21} \sqcap G_{11}) \rightarrow (G_{22} \sqcap G_{12})$

By the induction hypothesis on $G_{11} \sqsubseteq G_{21}$ and $G_{12} \sqsubseteq G_{22}$, we get that

- $G_{11} \sqcap G_{21} = G_{21} \sqcap G_{11} = G_{11}$
- $G_{12} \sqcap G_{22} = G_{22} \sqcap G_{12} = G_{12}$

Thus, we get that $(G_{11} \sqcap G_{21}) \rightarrow (G_{12} \sqcap G_{22}) = (G_{21} \sqcap G_{11}) \rightarrow (G_{22} \sqcap G_{12}) = G_{11} \rightarrow G_{12}$, and the result follows immediately.

Case ($G_{11} \times G_{12} \sqsubseteq G_{21} \times G_{22}$). Then, we know that

- $G_{11} \sqsubseteq G_{21}$ and $G_{12} \sqsubseteq G_{22}$
- $G_1 \sqcap G_2 = (G_{11} \times G_{12}) \sqcap (G_{21} \times G_{22}) = (G_{11} \sqcap G_{21}) \times (G_{12} \sqcap G_{22})$
- $G_2 \sqcap G_1 = (G_{21} \times G_{22}) \sqcap (G_{11} \times G_{12}) = (G_{21} \sqcap G_{11}) \times (G_{22} \sqcap G_{12})$

By the induction hypothesis on $G_{11} \sqsubseteq G_{21}$ and $G_{12} \sqsubseteq G_{22}$, we get that

- $G_{11} \sqcap G_{21} = G_{21} \sqcap G_{11} = G_{11}$
- $G_{12} \sqcap G_{22} = G_{22} \sqcap G_{12} = G_{12}$

Thus, we get that $(G_{11} \sqcap G_{21}) \times (G_{12} \sqcap G_{22}) = (G_{21} \sqcap G_{11}) \times (G_{22} \sqcap G_{12}) = G_{11} \times G_{12}$, and the result follows immediately.

Case ($\forall X.G_{11} \sqsubseteq \forall X.G_{21}$). Then, we know that

- $G_{11} \sqsubseteq G_{21}$
- $G_1 \sqcap G_2 = (\forall X.G_{11}) \sqcap (\forall X.G_{21}) = \forall X.(G_{11} \sqcap G_{21})$
- $G_2 \sqcap G_1 = (\forall X.G_{21}) \sqcap (\forall X.G_{11}) = \forall X.(G_{21} \sqcap G_{11})$

By the induction hypothesis on $G_{11} \sqsubseteq G_{21}$, we get that

- $G_{11} \sqcap G_{21} = G_{21} \sqcap G_{11} = G_{11}$

Thus, we get that $\forall X.(G_{11} \sqcap G_{21}) = \forall X.(G_{21} \sqcap G_{11}) = \forall X.G_{11}$, and the result follows immediately.

Case ($B \sqsubseteq ?_\delta$). Then, we get that $G_1 \sqcap G_2 = G_2 \sqcap G_1 = B \sqcap ?_\delta = ?_\delta \sqcap B = B = G_1$. Thus, the result follows immediately.

Case ($X \sqsubseteq ?_\delta$ and $X : F \in \delta$). Then, we get that $G_1 \sqcap G_2 = G_2 \sqcap G_1 = X \sqcap ?_\delta = ?_\delta \sqcap X = X = G_1$. Thus, the result follows immediately.

Case ($G_{11} \rightarrow G_{12} \sqsubseteq ?_\delta$). Then, we know that

- $G_{11} \sqsubseteq ?_\delta$ and $G_{12} \sqsubseteq ?_\delta$
- $G_1 \sqcap G_2 = (G_{11} \rightarrow G_{12}) \sqcap (?_\delta \rightarrow ?_\delta) = (G_{11} \sqcap ?_\delta) \rightarrow (G_{12} \sqcap ?_\delta)$
- $G_2 \sqcap G_1 = (?_\delta \rightarrow ?_\delta) \sqcap (G_{11} \rightarrow G_{12}) = (?_\delta \sqcap G_{11}) \rightarrow (?_\delta \sqcap G_{12})$

By the induction hypothesis on $G_{11} \sqsubseteq ?_\delta$ and $G_{12} \sqsubseteq ?_\delta$, we get that

- $G_{11} \sqcap ?_\delta = ?_\delta \sqcap G_{11} = G_{11}$
- $G_{12} \sqcap ?_\delta = ?_\delta \sqcap G_{12} = G_{12}$

Thus, we get that $(G_{11} \sqcap ?_\delta) \rightarrow (G_{12} \sqcap ?_\delta) = (?_\delta \sqcap G_{11}) \rightarrow (?_\delta \sqcap G_{12}) = G_{11} \rightarrow G_{12}$, and the result follows immediately.

Case ($G_{11} \times G_{12} \sqsubseteq ?_\delta$). Then, we know that

- $G_{11} \sqsubseteq ?_\delta$ and $G_{12} \sqsubseteq ?_\delta$
- $G_1 \sqcap G_2 = (G_{11} \times G_{12}) \sqcap (?_\delta \times ?_\delta) = (G_{11} \sqcap ?_\delta) \times (G_{12} \sqcap ?_\delta)$
- $G_2 \sqcap G_1 = (?_\delta \times ?_\delta) \sqcap (G_{11} \times G_{12}) = (?_\delta \sqcap G_{11}) \times (?_\delta \sqcap G_{12})$

By the induction hypothesis on $G_{11} \sqsubseteq ?_\delta$ and $G_{12} \sqsubseteq ?_\delta$, we get that

- $G_{11} \sqcap ?_\delta = ?_\delta \sqcap G_{11} = G_{11}$
- $G_{12} \sqcap ?_\delta = ?_\delta \sqcap G_{12} = G_{12}$

Thus, we get that $(G_{11} \sqcap ?_\delta) \times (G_{12} \sqcap ?_\delta) = (?_\delta \sqcap G_{11}) \times (?_\delta \sqcap G_{12}) = G_{11} \times G_{12}$, and the result follows immediately.

Case ($\forall X.G_{11} \sqsubseteq ?_\delta$). Then, we know that

- $G_{11} \sqsubseteq ?_{\delta, X: X}$
- $G_1 \sqcap G_2 = (\forall X.G_{11}) \sqcap ?_\delta = (\forall X.G_{11}) \sqcap (\forall X.?_{\delta, X: X}) = \forall X.(G_{11} \sqcap ?_{\delta, X: X})$
- $G_2 \sqcap G_1 = ?_\delta \sqcap (\forall X.G_{11}) = (\forall X.?_{\delta, X: X}) \sqcap (\forall X.G_{11}) = \forall X.(?_{\delta, X: X} \sqcap G_{11})$

By the induction hypothesis on $G_{11} \sqsubseteq ?_{\delta, X: X}$, we get that

- $G_{11} \sqcap ?_{\delta, X: X} = ?_{\delta, X: X} \sqcap G_{11} = G_{11}$

Thus, we get that $\forall X.(G_{11} \sqcap ?_{\delta, X:X}) = \forall X.(?_{\delta, X:X} \sqcap G_{11}) = \forall X.G_{11}$, and the result follows immediately.

Case ($?_{\delta} \sqsubseteq ?_{\delta'}$ and $\delta \subseteq \delta'$). Since $\delta \subseteq \delta'$, we know that $\delta \cap \delta' = \delta$. Then, we get that $G_1 \sqcap G_2 = G_2 \sqcap G_1 = ?_{\delta} \sqcap ?_{\delta'} = ?_{\delta'} \sqcap ?_{\delta} = ?_{\delta \cap \delta'} = ?_{\delta} = G_1$. Thus, the result follows immediately. □

Lemma B.118 *If* $G = G_1 \sqcap G_2$, *then* $G \sqsubseteq G_1$ *and* $G \sqsubseteq G_2$.

PROOF. By induction on $G_1 \sqcap G_2$.

Case ($B \sqcap B$). We know that $B \sqcap B = B$. Directly because $B \sqsubseteq B$.

Case ($X \sqcap X$). We know that $X \sqcap X = X$. Directly because $X \sqsubseteq X$.

Case ($(G'_1 \rightarrow G''_1) \sqcap (G'_2 \rightarrow G''_2)$). We know that $(G'_1 \rightarrow G''_1) \sqcap (G'_2 \rightarrow G''_2) = (G'_1 \sqcap G'_2) \rightarrow (G''_1 \sqcap G''_2)$. By the induction hypothesis on $(G'_1 \sqcap G'_2)$ and $(G''_1 \sqcap G''_2)$, we have that $(G'_1 \sqcap G'_2) \sqsubseteq G'_1$, $(G'_1 \sqcap G'_2) \sqsubseteq G'_2$, $(G''_1 \sqcap G''_2) \sqsubseteq G''_1$ and $(G''_1 \sqcap G''_2) \sqsubseteq G''_2$. Thus, by the $(\sqsubseteq_{\rightarrow})$ rule, we get that $(G'_1 \rightarrow G''_1) \sqcap (G'_2 \rightarrow G''_2) \sqsubseteq G'_1 \rightarrow G''_1$ and $(G'_1 \rightarrow G''_1) \sqcap (G'_2 \rightarrow G''_2) \sqsubseteq G'_2 \rightarrow G''_2$, as we are required to show.

Case ($(G'_1 \times G''_1) \sqcap (G'_2 \times G''_2)$). We know that $(G'_1 \times G''_1) \sqcap (G'_2 \times G''_2) = (G'_1 \sqcap G'_2) \times (G''_1 \sqcap G''_2)$. By the induction hypothesis on $(G'_1 \sqcap G'_2)$ and $(G''_1 \sqcap G''_2)$, we have that $(G'_1 \sqcap G'_2) \sqsubseteq G'_1$, $(G'_1 \sqcap G'_2) \sqsubseteq G'_2$, $(G''_1 \sqcap G''_2) \sqsubseteq G''_1$ and $(G''_1 \sqcap G''_2) \sqsubseteq G''_2$. Thus, by the (\sqsubseteq_{\times}) rule, we get that $(G'_1 \times G''_1) \sqcap (G'_2 \times G''_2) \sqsubseteq G'_1 \times G''_1$ and $(G'_1 \times G''_1) \sqcap (G'_2 \times G''_2) \sqsubseteq G'_2 \times G''_2$, as we are required to show.

Case ($(\forall X.G'_1) \sqcap (\forall X.G'_2)$). We know that $(\forall X.G'_1) \sqcap (\forall X.G'_2) = \forall X.(G'_1 \sqcap G'_2)$. By the induction hypothesis on $(G'_1 \sqcap G'_2)$, we have that $(G'_1 \sqcap G'_2) \sqsubseteq G'_1$ and $(G'_1 \sqcap G'_2) \sqsubseteq G'_2$. Thus, by the (\sqsubseteq_{\forall}) rule, we get that $(\forall X.G'_1) \sqcap (\forall X.G'_2) \sqsubseteq \forall X.G'_1$ and $(\forall X.G'_1) \sqcap (\forall X.G'_2) \sqsubseteq \forall X.G'_2$, as we are required to show.

Case ($?_{\delta_1} \sqcap ?_{\delta_2}$). We know that $?_{\delta_1} \sqcap ?_{\delta_2} = ?_{\delta_1 \cap \delta_2}$. We are required to prove that $?_{\delta_1 \cap \delta_2} \sqsubseteq ?_{\delta_1}$ and $?_{\delta_1 \cap \delta_2} \sqsubseteq ?_{\delta_2}$, which follows by rule $\sqsubseteq_{?}$, $\delta_1 \cap \delta_2 \subseteq \delta_1$ and $\delta_1 \cap \delta_2 \subseteq \delta_2$.

Case ($B \sqcap ?_{\delta} = B$). We know that $B \sqcap ?_{\delta} = B$. We are required to show that $B \sqsubseteq B$ and $B \sqsubseteq ?_{\delta}$, which follow immediately by rules (\sqsubseteq_B) and $(\sqsubseteq_{B?})$.

Case ($X \sqcap ?_{\delta} = X$). We know that $X \sqcap ?_{\delta} = X$ and $X : F \in \delta$. We are required to show that $X \sqsubseteq X$ and $X \sqsubseteq ?_{\delta}$, which follow immediately by rules (\sqsubseteq_X) , $(\sqsubseteq_{X?})$ and $X : F \in \delta$.

Case ($(G'_1 \rightarrow G''_1) \sqcap ?_{\delta}$). We know that $(G'_1 \rightarrow G''_1) \sqcap ?_{\delta} = (G'_1 \sqcap ?_{\delta}) \rightarrow (G''_1 \sqcap ?_{\delta})$. By the induction hypothesis on $(G'_1 \sqcap ?_{\delta})$ and $(G''_1 \sqcap ?_{\delta})$, we know that $(G'_1 \sqcap ?_{\delta}) \sqsubseteq G'_1$, $(G'_1 \sqcap ?_{\delta}) \sqsubseteq ?_{\delta}$, $(G''_1 \sqcap ?_{\delta}) \sqsubseteq G''_1$ and $(G''_1 \sqcap ?_{\delta}) \sqsubseteq ?_{\delta}$. We are required to show that $(G'_1 \sqcap ?_{\delta}) \rightarrow (G''_1 \sqcap ?_{\delta}) \sqsubseteq G'_1 \rightarrow G''_1$, or what is the same, by $(\sqsubseteq_{\rightarrow})$, we are required to prove that $(G'_1 \sqcap ?_{\delta}) \sqsubseteq G'_1$ and $(G''_1 \sqcap ?_{\delta}) \sqsubseteq G''_1$, which we already know. Also, we are required to prove that $(G'_1 \sqcap ?_{\delta}) \rightarrow (G''_1 \sqcap ?_{\delta}) \sqsubseteq ?_{\delta}$. Or what is the same, by $(\sqsubseteq_{\rightarrow?})$ rule, we are required to prove that $(G'_1 \sqcap ?_{\delta}) \sqsubseteq ?_{\delta}$ and $(G''_1 \sqcap ?_{\delta}) \sqsubseteq ?_{\delta}$, which we already know.

Case ($(G'_1 \times G''_1) \sqcap ?_{\delta}$). We know that $(G'_1 \times G''_1) \sqcap ?_{\delta} = (G'_1 \sqcap ?_{\delta}) \times (G''_1 \sqcap ?_{\delta})$. By the induction hypothesis on $(G'_1 \sqcap ?_{\delta})$ and $(G''_1 \sqcap ?_{\delta})$, we know that $(G'_1 \sqcap ?_{\delta}) \sqsubseteq G'_1$, $(G'_1 \sqcap ?_{\delta}) \sqsubseteq ?_{\delta}$, $(G''_1 \sqcap ?_{\delta}) \sqsubseteq G''_1$ and $(G''_1 \sqcap ?_{\delta}) \sqsubseteq ?_{\delta}$. We are required to show that $(G'_1 \sqcap ?_{\delta}) \times (G''_1 \sqcap ?_{\delta}) \sqsubseteq G'_1 \times G''_1$, or what is the same, by (\sqsubseteq_{\times}) , we are required to prove that $(G'_1 \sqcap ?_{\delta}) \sqsubseteq G'_1$ and $(G''_1 \sqcap ?_{\delta}) \sqsubseteq G''_1$, which we already know. Also, we are required to prove that $(G'_1 \sqcap ?_{\delta}) \times (G''_1 \sqcap ?_{\delta}) \sqsubseteq ?_{\delta}$. Or what is the same, by $(\sqsubseteq_{\times?})$ rule, we are required to prove that $(G'_1 \sqcap ?_{\delta}) \sqsubseteq ?_{\delta}$ and $(G''_1 \sqcap ?_{\delta}) \sqsubseteq ?_{\delta}$, which we already know.

Case ($(\forall X.G'_1) \sqcap ?_{\delta}$). We know that $(\forall X.G'_1) \sqcap ?_{\delta} = \forall X.(G'_1 \sqcap ?_{\delta, X:X})$. By the induction hypothesis on $(G'_1 \sqcap ?_{\delta, X:X})$, we know that $(G'_1 \sqcap ?_{\delta, X:X}) \sqsubseteq G'_1$, $(G'_1 \sqcap ?_{\delta, X:X}) \sqsubseteq ?_{\delta, X:X}$. We are required to show that $\forall X.(G'_1 \sqcap ?_{\delta, X:X}) \sqsubseteq \forall X.G'_1$, or what is the same, by (\sqsubseteq_{\forall}) , we are required to prove that $(G'_1 \sqcap ?_{\delta, X:X}) \sqsubseteq G'_1$, which we already know. Also, we are required to prove that $\forall X.(G'_1 \sqcap ?_{\delta, X:X}) \sqsubseteq ?_{\delta}$. Or what is

the same, by ($\sqsubseteq_{\forall?}$) rule, we are required to prove that $(G'_1 \sqcap ?_{\delta, X: X}) \sqsubseteq ?_{\delta}$ and $(G''_1 \sqcap ?_{\delta, X: X}) \sqsubseteq ?_{\delta, X: X}$, which we already know.

Case $(?_{\delta} \sqcap B)$. Analogous to the inverse case.

Case $(?_{\delta} \sqcap X)$. Analogous to the inverse case.

Case $(?_{\delta} \sqcap G'_2 \rightarrow G''_2)$. Analogous to the inverse case.

Case $(?_{\delta} \sqcap G'_2 \times G''_2)$. Analogous to the inverse case.

Case $(?_{\delta} \sqcap \forall X. G'_2)$. Analogous to the inverse case.

□

Lemma B.119 *If* $G_1 \sqsubseteq G_2$ *and* $G_2 \sqsubseteq G_3$, *then* $G_1 \sqsubseteq G_3$.

PROOF. Since $G_1 \sqsubseteq G_2$ and $G_2 \sqsubseteq G_3$, by Definition B.94, we know that $\gamma(G_1) \subseteq \gamma(G_2)$ and $\gamma(G_2) \subseteq \gamma(G_3)$. Therefore, we know that $\gamma(G_1) \subseteq \gamma(G_3)$. By Definition B.94, we get that $G_1 \sqsubseteq G_3$, as we are required to show. □

Lemma B.120 *If* $G_1 \sqsubseteq G_2$, $G'_1 \sqsubseteq G'_2$, $G_1 \sqcap G'_1$ *is defined*, *then* $G_2 \sqcap G'_2$ *is defined* *and* $(G_1 \sqcap G'_1) \sqsubseteq (G_2 \sqcap G'_2)$.

PROOF. We know by Lemma 7.8, that $G_1 \sqcap G'_1 = \alpha(\gamma(G_1) \cap \gamma(G'_1))$ is define, i.e $\gamma(G_1) \cap \gamma(G'_1) \neq \emptyset$. By Definition B.94, we know that $\gamma(G_1) \subseteq \gamma(G_2)$ and $\gamma(G'_1) \subseteq \gamma(G'_2)$, therefore $\gamma(G_2) \cap \gamma(G'_2) \neq \emptyset$, and this means that $\alpha(\gamma(G_2) \cap \gamma(G'_2))$ is defined and $G_2 \sqcap G'_2$ is defined. Also, we are required to prove that $(G_1 \sqcap G'_1) \sqsubseteq (G_2 \sqcap G'_2)$, or what is the same, by Definition B.94, we are required to show that $\alpha(\gamma(G_1) \cap \gamma(G'_1)) \sqsubseteq G_2 \sqcap G'_2$. We know that $\gamma(G_1) \cap \gamma(G'_1) \subseteq \gamma(G_2) \cap \gamma(G'_2)$ and by Soundness (Lemma B.92), we know that $\gamma(G_2) \cap \gamma(G'_2) \subseteq \gamma(\alpha(\gamma(G_2) \cap \gamma(G'_2)))$. Therefore, we get that $\gamma(G_1) \cap \gamma(G'_1) \subseteq \gamma(\alpha(\gamma(G_2) \cap \gamma(G'_2))) = \gamma(G_2 \sqcap G'_2)$. By Optimality (Lemma B.93), we get that $\alpha(\gamma(G_1) \cap \gamma(G'_1)) \sqsubseteq G_2 \sqcap G'_2$, as we are required to show. □

Lemma B.121 *If* $\text{initEv}(G_1, G_2) = \varepsilon$ *then* $\{(G_1 \sqcap G_2, \text{initPT}(G_1 \sqcap G_2, G_1), \text{initPT}(G_1 \sqcap G_2, G_2))\} = \varepsilon$.

PROOF. We proceed by induction on the syntax of $\text{initEv}(G_1, G_2)$.

Case $(\text{initEv}(B, B))$. $G_1 = B$ and $G_2 = B$. We know that

- $\text{initEv}(G_1, G_2) = \text{initEv}(B, B) = \text{refl}_B^{\sqsubseteq} = \{(B, B, B)\}$
- $G_1 \sqcap G_2 = B \sqcap B = B$
- $[(G_1 \sqcap G_2, \text{initPT}(G_1 \sqcap G_2, G_1), \text{initPT}(G_1 \sqcap G_2, G_2))] = [(B \sqcap B, \text{initPT}(B \sqcap B, B), \text{initPT}(B \sqcap B, B))] = [(B, \text{initPT}(B, B), \text{initPT}(B, B))] = \{(B, B, B)\}$

Therefore, the result follows immediately.

Case $(\text{initEv}(X, X))$. $G_1 = X$ and $G_2 = X$. We know that

- $\text{initEv}(G_1, G_2) = \text{initEv}(X, X) = \text{refl}_X^{\sqsubseteq} = \{(X, X, X)\}$

- $G_1 \sqcap G_2 = X \sqcap X = X$
- $[(G_1 \sqcap G_2, \text{initPT}(G_1 \sqcap G_2, G_1), \text{initPT}(G_1 \sqcap G_2, G_2))] = [(X \sqcap X, \text{initPT}(X \sqcap X, X), \text{initPT}(X \sqcap X, X))] = [(X, \text{initPT}(X, X), \text{initPT}(X, X))] = \{(X, X, X)\}$

Therefore, the result follows immediately.

Case $(\text{initEv}(G_{11} \rightarrow G_{12}, G_{21} \rightarrow G_{22}))$. $G_1 = G_{11} \rightarrow G_{12}$ and $G_2 = G_{21} \rightarrow G_{22}$. We know that

- $\text{initEv}(G_1, G_2) = \text{initEv}(G_{11} \rightarrow G_{12}, G_{21} \rightarrow G_{22}) = \text{initEv}(G_{11}, G_{21}) \rightarrow \text{initEv}(G_{12}, G_{22})$
- $G \rightarrow G' = G_1 \sqcap G_2 = (G_{11} \rightarrow G_{12}) \sqcap (G_{21} \rightarrow G_{22}) = (G_{11} \sqcap G_{21}) \rightarrow (G_{12} \sqcap G_{22})$
- $c_1 \rightarrow c_1 = \text{initPT}(G_1 \sqcap G_2, G_1) = \text{initPT}((G_{11} \sqcap G_{21}) \rightarrow (G_{12} \sqcap G_{22}), G_{11} \rightarrow G_{12}) = \text{initPT}((G_{11} \sqcap G_{21}), G_{11}) \rightarrow \text{initPT}((G_{12} \sqcap G_{22}), G_{12})$
- $c_2 \rightarrow c_2' = \text{initPT}(G_1 \sqcap G_2, G_2) = \text{initPT}((G_{11} \sqcap G_{21}) \rightarrow (G_{12} \sqcap G_{22}), G_{21} \rightarrow G_{22}) = \text{initPT}((G_{11} \sqcap G_{21}), G_{21}) \rightarrow \text{initPT}((G_{12} \sqcap G_{22}), G_{22})$
- $[(G_1 \sqcap G_2, \text{initPT}(G_1 \sqcap G_2, G_1), \text{initPT}(G_1 \sqcap G_2, G_2))] = [(G \rightarrow G', c_1 \rightarrow c_1, c_2 \rightarrow c_2')]$

By the induction hypothesis on $\text{initEv}(G_{11}, G_{21})$ and $\text{initEv}(G_{12}, G_{22})$, we get that

- $\text{initEv}(G_{11}, G_{21}) = [((G_{11} \sqcap G_{21}), \text{initPT}((G_{11} \sqcap G_{21}), G_{11}), \text{initPT}((G_{11} \sqcap G_{21}), G_{21})) = [(G, c_1, c_2)]$
- $\text{initEv}(G_{12}, G_{22}) = [((G_{12} \sqcap G_{22}), \text{initPT}((G_{12} \sqcap G_{22}), G_{12}), \text{initPT}((G_{12} \sqcap G_{22}), G_{22})) = [(G', c_1, c_2')]$

Thus, we get that $\text{initEv}(G_1, G_2) = \text{initEv}(G_{11}, G_{21}) \rightarrow \text{initEv}(G_{12}, G_{22}) = [(G \rightarrow G', c_1 \rightarrow c_1, c_2 \rightarrow c_2')]$. Therefore, the result follows immediately.

Case $(\text{initEv}(G_{11} \times G_{12}, G_{21} \times G_{22}))$. $G_1 = G_{11} \times G_{12}$ and $G_2 = G_{21} \times G_{22}$. We know that

- $\text{initEv}(G_1, G_2) = \text{initEv}(G_{11} \times G_{12}, G_{21} \times G_{22}) = \text{initEv}(G_{11}, G_{21}) \times \text{initEv}(G_{12}, G_{22})$
- $G \times G' = G_1 \sqcap G_2 = (G_{11} \times G_{12}) \sqcap (G_{21} \times G_{22}) = (G_{11} \sqcap G_{21}) \times (G_{12} \sqcap G_{22})$
- $c_1 \times c_1 = \text{initPT}(G_1 \sqcap G_2, G_1) = \text{initPT}((G_{11} \sqcap G_{21}) \times (G_{12} \sqcap G_{22}), G_{11} \times G_{12}) = \text{initPT}((G_{11} \sqcap G_{21}), G_{11}) \times \text{initPT}((G_{12} \sqcap G_{22}), G_{12})$
- $c_2 \times c_2' = \text{initPT}(G_1 \sqcap G_2, G_2) = \text{initPT}((G_{11} \sqcap G_{21}) \times (G_{12} \sqcap G_{22}), G_{21} \times G_{22}) = \text{initPT}((G_{11} \sqcap G_{21}), G_{21}) \times \text{initPT}((G_{12} \sqcap G_{22}), G_{22})$
- $[(G_1 \sqcap G_2, \text{initPT}(G_1 \sqcap G_2, G_1), \text{initPT}(G_1 \sqcap G_2, G_2))] = [(G \times G', c_1 \times c_1, c_2 \times c_2')]$

By the induction hypothesis on $\text{initEv}(G_{11}, G_{21})$ and $\text{initEv}(G_{12}, G_{22})$, we get that

- $\text{initEv}(G_{11}, G_{21}) = [((G_{11} \sqcap G_{21}), \text{initPT}((G_{11} \sqcap G_{21}), G_{11}), \text{initPT}((G_{11} \sqcap G_{21}), G_{21})) = [(G, c_1, c_2)]$

- $\text{initEv}(\mathbf{G}_{12}, \mathbf{G}_{22}) = [((\mathbf{G}_{12} \sqcap \mathbf{G}_{22}), \text{initPT}((\mathbf{G}_{12} \sqcap \mathbf{G}_{22}), \mathbf{G}_{12}), \text{initPT}((\mathbf{G}_{12} \sqcap \mathbf{G}_{22}), \mathbf{G}_{22})) = [(\mathbf{G}', \mathbf{c}_1, \mathbf{c}'_2)]$

Thus, we get that $\text{initEv}(\mathbf{G}_1, \mathbf{G}_2) = \text{initEv}(\mathbf{G}_{11}, \mathbf{G}_{21}) \times \text{initEv}(\mathbf{G}_{12}, \mathbf{G}_{22}) = [(\mathbf{G} \times \mathbf{G}', \mathbf{c}_1 \times \mathbf{c}_1, \mathbf{c}_2 \times \mathbf{c}'_2)]$. Therefore, the result follows immediately.

Case $(\text{initEv}(\forall \mathbf{X}.\mathbf{G}_{11}, \forall \mathbf{X}.\mathbf{G}_{21}))$. $\mathbf{G}_1 = \forall \mathbf{X}.\mathbf{G}_{11}$ and $\mathbf{G}_2 = \forall \mathbf{X}.\mathbf{G}_{21}$. We know that

- $\text{initEv}(\mathbf{G}_1, \mathbf{G}_2) = \text{initEv}(\forall \mathbf{X}.\mathbf{G}_{11}, \forall \mathbf{X}.\mathbf{G}_{21}) = \forall \mathbf{X}.\text{initEv}(\mathbf{G}_{11}, \mathbf{G}_{21})$
- $\forall \mathbf{X}.\mathbf{G} = \mathbf{G}_1 \sqcap \mathbf{G}_2 = (\forall \mathbf{X}.\mathbf{G}_{11}) \sqcap (\forall \mathbf{X}.\mathbf{G}_{21}) = \forall \mathbf{X}.\mathbf{G}_{11} \sqcap \mathbf{G}_{21}$
- $\forall \mathbf{X}.\mathbf{c}_1 = \text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_1) = \text{initPT}(\forall \mathbf{X}.\mathbf{G}_{11} \sqcap \mathbf{G}_{21}, \forall \mathbf{X}.\mathbf{G}_{11}) = \forall \mathbf{X}.\text{initPT}(\mathbf{G}_{11} \sqcap \mathbf{G}_{21}, \mathbf{G}_{11})$
- $\forall \mathbf{X}.\mathbf{c}_2 = \text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_2) = \text{initPT}(\forall \mathbf{X}.\mathbf{G}_{11} \sqcap \mathbf{G}_{21}, \forall \mathbf{X}.\mathbf{G}_{21}) = \forall \mathbf{X}.\text{initPT}(\mathbf{G}_{11} \sqcap \mathbf{G}_{21}, \mathbf{G}_{21})$
- $[[(\mathbf{G}_1 \sqcap \mathbf{G}_2, \text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_1), \text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_2))] = [(\forall \mathbf{X}.\mathbf{G}, \forall \mathbf{X}.\mathbf{c}_1, \forall \mathbf{X}.\mathbf{c}_2)]$

By the induction hypothesis on $\text{initEv}(\mathbf{G}_{11}, \mathbf{G}_{21})$, we get that

- $\text{initEv}(\mathbf{G}_{11}, \mathbf{G}_{21}) = [((\mathbf{G}_{11} \sqcap \mathbf{G}_{21}), \text{initPT}((\mathbf{G}_{11} \sqcap \mathbf{G}_{21}), \mathbf{G}_{11}), \text{initPT}((\mathbf{G}_{11} \sqcap \mathbf{G}_{21}), \mathbf{G}_{21})) = [(\mathbf{G}, \mathbf{c}_1, \mathbf{c}_2)]$

Thus, we get that $\text{initEv}(\mathbf{G}_1, \mathbf{G}_2) = \forall \mathbf{X}.\text{initEv}(\mathbf{G}_{11}, \mathbf{G}_{21}) = [(\forall \mathbf{X}.\mathbf{G}, \forall \mathbf{X}.\mathbf{c}_1, \forall \mathbf{X}.\mathbf{c}_2)]$. Therefore, the result follows immediately.

Case $(\text{initEv}(\mathbf{G}_1, ?_\delta)$ and $\text{ftv}(\mathbf{G}_1) \in \delta)$. $\mathbf{G}_2 = ?_\delta$.

- $\text{initEv}(\mathbf{G}_1, \mathbf{G}_2) = \text{initEv}(\mathbf{G}_1, ?_\delta) = [(\mathbf{G}_1, \text{refl}_{\mathbf{G}_1}^{\sqsubseteq}, \text{inject}_{\mathbf{G}_1}^{\sqsubseteq})]$.
- $\mathbf{G}_1 \sqsubseteq ?_\delta$, by Lemma B.100 and $\text{ftv}(\mathbf{G}_1) \in \delta$.
- $\mathbf{G}_1 \sqcap ?_\delta = \mathbf{G}_1$, by Lemma B.117 and $\mathbf{G}_1 \sqsubseteq ?_\delta$.
- $\text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_1) = \text{initPT}(\mathbf{G}_1 \sqcap ?_\delta, \mathbf{G}_1) = \text{initPT}(\mathbf{G}_1, \mathbf{G}_1) = \text{refl}_{\mathbf{G}_1}^{\sqsubseteq}$, by Lemma B.114.
- $\text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_2) = \text{initPT}(\mathbf{G}_1 \sqcap ?_\delta, ?_\delta) = \text{initPT}(\mathbf{G}_1, ?_\delta) = \text{inject}_{\mathbf{G}_1}^{\sqsubseteq}$, by Lemma B.115.
- $[[(\mathbf{G}_1 \sqcap \mathbf{G}_2, \text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_1), \text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_2))] = [(\mathbf{G}_1, \text{refl}_{\mathbf{G}_1}^{\sqsubseteq}, \text{inject}_{\mathbf{G}_1}^{\sqsubseteq})]$.

Therefore, the result follows immediately.

Case $(\text{initEv}(?_\delta, \mathbf{G}_2)$ and $\text{ftv}(\mathbf{G}_2) \in \delta)$. $\mathbf{G}_1 = ?_\delta$.

- $\text{initEv}(\mathbf{G}_1, \mathbf{G}_2) = \text{initEv}(?_\delta, \mathbf{G}_2) = [(\mathbf{G}_2, \text{inject}_{\mathbf{G}_2}^{\sqsubseteq}, \text{refl}_{\mathbf{G}_2}^{\sqsubseteq})]$.
- $\mathbf{G}_2 \sqsubseteq ?_\delta$, by Lemma B.100 and $\text{ftv}(\mathbf{G}_2) \in \delta$.
- $\mathbf{G}_2 \sqcap ?_\delta = \mathbf{G}_2$, by Lemma B.117 and $\mathbf{G}_2 \sqsubseteq ?_\delta$.

- $\text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_1) = \text{initPT}(\mathbf{?}_\delta \sqcap \mathbf{G}_2, \mathbf{?}_\delta) = \text{initPT}(\mathbf{G}_2, \mathbf{?}_\delta) = \text{inject}_{\mathbf{G}_2}^{\sqsubseteq}$, by Lemma B.115.
- $\text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_2) = \text{initPT}(\mathbf{?}_\delta \sqcap \mathbf{G}_2, \mathbf{G}_2) = \text{initPT}(\mathbf{G}_2, \mathbf{G}_2) = \text{refl}_{\mathbf{G}_2}^{\sqsubseteq}$, by Lemma B.114.
- $[(\mathbf{G}_1 \sqcap \mathbf{G}_2, \text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_1), \text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}_2, \mathbf{G}_2))] = [(\mathbf{G}_2, \text{inject}_{\mathbf{G}_2}^{\sqsubseteq}, \text{refl}_{\mathbf{G}_2}^{\sqsubseteq})]$.

Therefore, the result follows immediately. □

Lemma B.122 *If $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, $\varepsilon_1 = \text{reflEv}(\mathbf{G}_1)$, $\varepsilon_2 = \text{reflEv}(\mathbf{G}_2)$ and $\mathbf{c} = \text{initPT}(\mathbf{G}_1, \mathbf{G}_2)$, then $[\mathbf{c}]_{\varepsilon_1} \sqsubseteq \varepsilon_2[\mathbf{c}]$.*

PROOF. We know that $\varepsilon_1 = \text{reflEv}(\mathbf{G}_1) = \text{initEv}(\mathbf{G}_1, \mathbf{G}_1) = \{(\mathbf{G}_1, \text{refl}_{\mathbf{G}_1}^{\sqsubseteq}, \text{refl}_{\mathbf{G}_1}^{\sqsubseteq})\}$ and $\varepsilon_2 = \text{reflEv}(\mathbf{G}_2) = \text{initEv}(\mathbf{G}_2, \mathbf{G}_2) = \{(\mathbf{G}_2, \text{refl}_{\mathbf{G}_2}^{\sqsubseteq}, \text{refl}_{\mathbf{G}_2}^{\sqsubseteq})\}$, by Lemma B.63. Since $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, we know that $\text{initPT}(\mathbf{G}_1, \mathbf{G}_1)$ is defined. If we take $\mathbf{c}_t = \text{initPT}(\mathbf{G}_1, \mathbf{G}_1)$, by Lemma B.87, we get that

- $\text{initPT}(\mathbf{G}_1, \mathbf{G}_2); \text{refl}_{\mathbf{G}_1}^{\sqsubseteq} = \text{initPT}(\mathbf{G}_1, \mathbf{G}_2)$
- $\text{refl}_{\mathbf{G}_1}^{\sqsubseteq}; \text{initPT}(\mathbf{G}_1, \mathbf{G}_2) = \text{initPT}(\mathbf{G}_1, \mathbf{G}_2)$
- $\text{initPT}(\mathbf{G}_1, \mathbf{G}_2); \text{refl}_{\mathbf{G}_2}^{\sqsubseteq} = \text{initPT}(\mathbf{G}_1, \mathbf{G}_2)$
- $\text{refl}_{\mathbf{G}_2}^{\sqsubseteq}; \text{initPT}(\mathbf{G}_1, \mathbf{G}_2) = \text{initPT}(\mathbf{G}_1, \mathbf{G}_2)$

as we are required to show. □

Lemma B.123 *If $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$ and $\mathbf{G}_2 \sqsubseteq \mathbf{G}_3$, then $\text{initPT}(\mathbf{G}_1, \mathbf{G}_2); \text{initPT}(\mathbf{G}_2, \mathbf{G}_3) = \text{initPT}(\mathbf{G}_1, \mathbf{G}_3)$.*

PROOF. Since $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$ and $\mathbf{G}_2 \sqsubseteq \mathbf{G}_3$. We know that $\text{initPT}(\mathbf{G}_1, \mathbf{G}_2) : \mathbf{G}_1 \sqsubseteq \mathbf{G}_2$ and $\text{initPT}(\mathbf{G}_2, \mathbf{G}_3) : \mathbf{G}_2 \sqsubseteq \mathbf{G}_3$. Thus, by Lemma B.20, we get that $\text{initPT}(\mathbf{G}_1, \mathbf{G}_2); \text{initPT}(\mathbf{G}_2, \mathbf{G}_3) = \mathbf{c}_3$, such that $\mathbf{c}_3 : \mathbf{G}_1 \sqsubseteq \mathbf{G}_3$. Therefore, by Lemma B.112, we get that $\mathbf{c}_3 = \text{initPT}(\mathbf{G}_1, \mathbf{G}_3)$, and the result holds. □

Lemma B.124 *If $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, $\mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$, $\varepsilon_1 = \text{initEv}(\mathbf{G}_1, \mathbf{G}'_1)$, $\varepsilon_2 = \text{initEv}(\mathbf{G}_2, \mathbf{G}'_2)$, $\mathbf{c} = \text{initPT}(\mathbf{G}_1, \mathbf{G}_2)$ and $\mathbf{c}' = \text{initPT}(\mathbf{G}'_1, \mathbf{G}'_2)$, then $[\mathbf{c}]_{\varepsilon_1} \sqsubseteq \varepsilon_2[\mathbf{c}']$.*

PROOF. We know that

- $\varepsilon_1 = \text{initEv}(\mathbf{G}_1, \mathbf{G}'_1) = [(\mathbf{G}_1 \sqcap \mathbf{G}'_1, \text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}'_1, \mathbf{G}_1), \text{initPT}(\mathbf{G}_1 \sqcap \mathbf{G}'_1, \mathbf{G}'_1))] = [(\mathbf{G}'_1, \text{initPT}(\mathbf{G}'_1, \mathbf{G}_1), \text{initPT}(\mathbf{G}'_1, \mathbf{G}'_1))]$, where $\mathbf{G}_1 \sqcap \mathbf{G}'_1 = \mathbf{G}'_1$.
- $\varepsilon_2 = \text{initEv}(\mathbf{G}_2, \mathbf{G}'_2) = [(\mathbf{G}_2 \sqcap \mathbf{G}'_2, \text{initPT}(\mathbf{G}_2 \sqcap \mathbf{G}'_2, \mathbf{G}_2), \text{initPT}(\mathbf{G}_2 \sqcap \mathbf{G}'_2, \mathbf{G}'_2))] = [(\mathbf{G}'_2, \text{initPT}(\mathbf{G}'_2, \mathbf{G}_2), \text{initPT}(\mathbf{G}'_2, \mathbf{G}'_2))]$, where $\mathbf{G}_2 \sqcap \mathbf{G}'_2 = \mathbf{G}'_2$.
- Since $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, $\mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$, $\mathbf{G}_1 \sqcap \mathbf{G}'_1$ and $\mathbf{G}_2 \sqcap \mathbf{G}'_2$ are defined, by Lemma B.120, we get that $\mathbf{G}_1 \sqcap \mathbf{G}'_1 = \mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2 = \mathbf{G}_2 \sqcap \mathbf{G}'_2$.

- $G_1'' \sqsubseteq G_1$, $G_1' \sqsubseteq G_1'$, $G_2'' \sqsubseteq G_2$ and $G_2' \sqsubseteq G_2'$, by Lemma B.118. Therefore, $G_1'' \sqsubseteq G_1 \sqsubseteq G_2$, $G_1' \sqsubseteq G_1' \sqsubseteq G_2'$, $G_1'' \sqsubseteq G_2'' \sqsubseteq G_2$ and $G_1' \sqsubseteq G_2' \sqsubseteq G_2'$, by Lemma B.119.

Since $G_1'' \sqsubseteq G_1 \sqsubseteq G_2$, $G_1' \sqsubseteq G_1' \sqsubseteq G_2'$, $G_1'' \sqsubseteq G_2'' \sqsubseteq G_2$ and $G_1' \sqsubseteq G_2' \sqsubseteq G_2'$, by Lemma B.123, we get that

- $\text{initPT}(G_1'', G_2''); \text{initPT}(G_2'', G_2) = \text{initPT}(G_1'', G_2)$
- $\text{initPT}(G_1'', G_1); \text{initPT}(G_1, G_2) = \text{initPT}(G_1'', G_2)$
- $\text{initPT}(G_1'', G_2''); \text{initPT}(G_2'', G_2') = \text{initPT}(G_1'', G_2')$
- $\text{initPT}(G_1'', G_1'); \text{initPT}(G_1', G_2') = \text{initPT}(G_1'', G_2')$

We are required to prove that $[c]_{\varepsilon_1} \sqsubseteq \varepsilon_2[c']$, where $c = \text{initPT}(G_1, G_2)$ and $c' = \text{initPT}(G_1', G_2')$. By Definition 6.11, if we take $c_t = \text{initPT}(G_1'', G_2'')$, $c_1' = \text{initPT}(G_1'', G_2)$ and $c_2' = \text{initPT}(G_1', G_2')$, the result follows immediately.

□

Lemma B.125 *If $G_1 \leq G_2$, $\varepsilon_1 = \text{instEv}(G_1, X, F)$, $\varepsilon_2 = \text{instEv}(G_2, X, F)$, $c = \text{initPT}(G_1, G_2)$, $c' = \text{initPT}(G_1[F/X] \setminus X, G_2[F/X] \setminus X)$, then $[c[F/X]]_{\varepsilon_1} \sqsubseteq \varepsilon_2[c']$.*

PROOF. We proceed by induction on $G_1 \leq G_2$.

Case ($B \leq B$). $G_1 = B$ and $G_2 = B$. We know that

- $\varepsilon_1 = \text{instEv}(G_1, X, F) = \text{instEv}(B, X, F) = \text{reflEv}(B) = \{(B, B, B)\}$
- $\varepsilon_2 = \text{instEv}(G_2, X, F) = \text{instEv}(B, X, F) = \text{reflEv}(B) = \{(B, B, B)\}$
- $c = \text{initPT}(G_1, G_2) = \text{initPT}(B, B) = B$
- $c' = \text{initPT}(G_1[F/X] \setminus X, G_2[F/X] \setminus X) = \text{initPT}(B[F/X] \setminus X, B[F/X] \setminus X) = \text{initPT}(B, B) = B$

Therefore, we are required to prove $[c[F/X]]_{\varepsilon_1} \sqsubseteq \varepsilon_2[c']$. Or what is the same, we are required to prove $[B]\{(B, B, B)\} \sqsubseteq \{(B, B, B)\}[B]$, because $c[F/X] = B_{F/X} = B$. Note that the result follows immediately.

Case ($Y \leq Y$). $G_1 = Y$ and $G_2 = Y$. We divide this case in two; if $Y = X$ or $Y \not\sim X$.

- $Y = X$. We know that
 - $\varepsilon_1 = \text{instEv}(G_1, X, F) = \text{instEv}(X, X, F) = \text{reflEv}(F) = [(F, F, F)]$
 - $\varepsilon_2 = \text{instEv}(G_2, X, F) = \text{instEv}(X, X, F) = \text{reflEv}(F) = [(F, F, F)]$
 - $c = \text{initPT}(G_1, G_2) = \text{initPT}(X, X) = X$
 - $c' = \text{initPT}(G_1[F/X] \setminus X, G_2[F/X] \setminus X) = \text{initPT}(X[F/X] \setminus X, X[F/X] \setminus X) = \text{initPT}(F, F) = F$

Therefore, we are required to prove $[c[F/X]]_{\varepsilon_1} \sqsubseteq \varepsilon_2[c']$. Or what is the same, we are required to prove $[F][(F, F, F)] \sqsubseteq [(F, F, F)][F]$, because $c[F/X] = X[F/X] = F$. Note that the result follows immediately.

- $Y \not\leftrightarrow X$. We know that
 - $\varepsilon_1 = \text{instEv}(G_1, X, F) = \text{instEv}(Y, X, F) = \text{reflEv}(Y) = [(Y, Y, Y)]$
 - $\varepsilon_2 = \text{instEv}(G_2, X, F) = \text{instEv}(Y, X, F) = \text{reflEv}(Y) = [(Y, Y, Y)]$
 - $c = \text{initPT}(G_1, G_2) = \text{initPT}(Y, Y) = Y$
 - $c' = \text{initPT}(G_1[F/X]\setminus X, G_2[F/X]\setminus X) = \text{initPT}(Y[F/X]\setminus X, Y[F/X]\setminus X) = \text{initPT}(Y, Y) = Y$

Therefore, we are required to prove $[c[F/X]]\varepsilon_1 \sqsubseteq \varepsilon_2[c']$. Or what is the same, we are required to prove $[Y][(Y, Y, Y)] \sqsubseteq [(Y, Y, Y)][Y]$, because $c[F/X] = Y[F/X] = Y$. Note that the result follows immediately.

Case ($G_{11} \rightarrow G_{12} \leq G_{21} \rightarrow G_{22}$). $G_1 = G_{11} \rightarrow G_{12}$ and $G_2 = G_{21} \rightarrow G_{22}$. We know that

- $G_{11} \leq G_{21}$ and $G_{12} \leq G_{22}$
- $\varepsilon_1 = \text{instEv}(G_1, X, F) = \text{instEv}(G_{11} \rightarrow G_{12}, X, F) = \text{instEv}(G_{11}, X, F) \rightarrow \text{instEv}(G_{12}, X, F)$
- $\varepsilon_2 = \text{instEv}(G_2, X, F) = \text{instEv}(G_{21} \rightarrow G_{22}, X, F) = \text{instEv}(G_{21}, X, F) \rightarrow \text{instEv}(G_{22}, X, F)$
- $c = c_1 \rightarrow c_2 = \text{initPT}(G_1, G_2) = \text{initPT}(G_{11} \rightarrow G_{12}, G_{21} \rightarrow G_{22}) = \text{initPT}(G_{11}, G_{21}) \rightarrow \text{initPT}(G_{12}, G_{22})$
- $c' = c_1 \rightarrow c'_2 = \text{initPT}(G_1[F/X]\setminus X, G_2[F/X]\setminus X) = \text{initPT}((G_{11} \rightarrow G_{12})[F/X]\setminus X, (G_{21} \rightarrow G_{22})[F/X]\setminus X) = \text{initPT}(G_{11}[F/X]\setminus X, G_{21}[F/X]\setminus X) \rightarrow \text{initPT}(G_{12}[F/X]\setminus X, G_{22}[F/X]\setminus X)$

Therefore, we are required to prove $[c[F/X]]\varepsilon_1 \sqsubseteq \varepsilon_2[c']$. Or what is the same, we are required to prove

$$[c_1[F/X] \rightarrow c_2[F/X]](\text{instEv}(G_{11}, X, F) \rightarrow \text{instEv}(G_{12}, X, F)) \sqsubseteq (\text{instEv}(G_{21}, X, F) \rightarrow \text{instEv}(G_{22}, X, F))[c_1 \rightarrow c'_2]$$

By the induction hypothesis on $G_{11} \leq G_{21}$ and $G_{12} \leq G_{22}$, we get that

- $[c_1[F/X]]\text{instEv}(G_{11}, X, F) \sqsubseteq \text{instEv}(G_{21}, X, F)[c_1]$
- $[c_2[F/X]]\text{instEv}(G_{12}, X, F) \sqsubseteq \text{instEv}(G_{22}, X, F)[c'_2]$

Thus, the result follows immediately, by Lemma B.74.

Case ($G_{11} \times G_{12} \leq G_{21} \times G_{22}$). $G_1 = G_{11} \times G_{12}$ and $G_2 = G_{21} \times G_{22}$. We know that

- $G_{11} \leq G_{21}$ and $G_{12} \leq G_{22}$
- $\varepsilon_1 = \text{instEv}(G_1, X, F) = \text{instEv}(G_{11} \times G_{12}, X, F) = \text{instEv}(G_{11}, X, F) \times \text{instEv}(G_{12}, X, F)$
- $\varepsilon_2 = \text{instEv}(G_2, X, F) = \text{instEv}(G_{21} \times G_{22}, X, F) = \text{instEv}(G_{21}, X, F) \times \text{instEv}(G_{22}, X, F)$
- $c = c_1 \times c_2 = \text{initPT}(G_1, G_2) = \text{initPT}(G_{11} \times G_{12}, G_{21} \times G_{22}) = \text{initPT}(G_{11}, G_{21}) \times \text{initPT}(G_{12}, G_{22})$

- $\mathbf{c}' = \mathbf{c}_1 \times \mathbf{c}'_2 = \text{initPT}(\mathbf{G}_1[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}, \mathbf{G}_2[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}) =$
 $\text{initPT}((\mathbf{G}_{11} \times \mathbf{G}_{12})[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}, (\mathbf{G}_{21} \times \mathbf{G}_{22})[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}) =$
 $\text{initPT}(\mathbf{G}_{11}[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}, \mathbf{G}_{21}[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}) \times \text{initPT}(\mathbf{G}_{12}[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}, \mathbf{G}_{22}[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X})$

Therefore, we are required to prove $[\mathbf{c}[\mathbf{F}/\mathbf{X}]]\varepsilon_1 \sqsubseteq \varepsilon_2[\mathbf{c}']$. Or what is the same, we are required to prove

$$[\mathbf{c}_1[\mathbf{F}/\mathbf{X}] \times \mathbf{c}'_2[\mathbf{F}/\mathbf{X}]](\text{instEv}(\mathbf{G}_{11}, \mathbf{X}, \mathbf{F}) \times \text{instEv}(\mathbf{G}_{12}, \mathbf{X}, \mathbf{F})) \sqsubseteq (\text{instEv}(\mathbf{G}_{21}, \mathbf{X}, \mathbf{F}) \times \text{instEv}(\mathbf{G}_{22}, \mathbf{X}, \mathbf{F}))[\mathbf{c}_1 \times \mathbf{c}'_2]$$

By the induction hypothesis on $\mathbf{G}_{11} \leq \mathbf{G}_{21}$ and $\mathbf{G}_{12} \leq \mathbf{G}_{22}$, we get that

- $[\mathbf{c}_1[\mathbf{F}/\mathbf{X}]]\text{instEv}(\mathbf{G}_{11}, \mathbf{X}, \mathbf{F}) \sqsubseteq \text{instEv}(\mathbf{G}_{21}, \mathbf{X}, \mathbf{F})[\mathbf{c}_1]$
- $[\mathbf{c}'_2[\mathbf{F}/\mathbf{X}]]\text{instEv}(\mathbf{G}_{12}, \mathbf{X}, \mathbf{F}) \sqsubseteq \text{instEv}(\mathbf{G}_{22}, \mathbf{X}, \mathbf{F})[\mathbf{c}'_2]$

Thus, the result follows immediately, by Lemma B.75.

Case $(\forall \mathbf{X}. \mathbf{G}_{11} \leq \forall \mathbf{X}. \mathbf{G}_{21})$. $\mathbf{G}_1 = \forall \mathbf{X}. \mathbf{G}_{11}$ and $\mathbf{G}_2 = \forall \mathbf{X}. \mathbf{G}_{21}$. We know that

- $\mathbf{G}_{11} \leq \mathbf{G}_{21}$
- $\varepsilon_1 = \text{instEv}(\mathbf{G}_1, \mathbf{X}, \mathbf{F}) = \text{instEv}(\forall \mathbf{X}. \mathbf{G}_{11}, \mathbf{X}, \mathbf{F}) = \forall \mathbf{X}. \text{instEv}(\mathbf{G}_{11}, \mathbf{X}, \mathbf{F})$
- $\varepsilon_2 = \text{instEv}(\mathbf{G}_2, \mathbf{X}, \mathbf{F}) = \text{instEv}(\forall \mathbf{X}. \mathbf{G}_{21}, \mathbf{X}, \mathbf{F}) = \forall \mathbf{X}. \text{instEv}(\mathbf{G}_{21}, \mathbf{X}, \mathbf{F})$
- $\mathbf{c} = \forall \mathbf{X}. \mathbf{c}_1 = \text{initPT}(\mathbf{G}_1, \mathbf{G}_2) = \text{initPT}(\forall \mathbf{X}. \mathbf{G}_{11}, \forall \mathbf{X}. \mathbf{G}_{21}) = \forall \mathbf{X}. \text{initPT}(\mathbf{G}_{11}, \mathbf{G}_{21})$
- $\mathbf{c}' = \forall \mathbf{X}. \mathbf{c}'_1 = \text{initPT}(\mathbf{G}_1[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}, \mathbf{G}_2[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}) = \text{initPT}((\forall \mathbf{X}. \mathbf{G}_{11})[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}, (\forall \mathbf{X}. \mathbf{G}_{21})[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}) = \forall \mathbf{X}. \text{initPT}(\mathbf{G}_{11}[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}, \mathbf{G}_{21}[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X})$

Therefore, we are required to prove $[\mathbf{c}[\mathbf{F}/\mathbf{X}]]\varepsilon_1 \sqsubseteq \varepsilon_2[\mathbf{c}']$. Or what is the same, we are required to prove

$$[\forall \mathbf{X}. (\mathbf{c}_1[\mathbf{F}/\mathbf{X}])](\forall \mathbf{X}. \text{instEv}(\mathbf{G}_{11}, \mathbf{X}, \mathbf{F})) \sqsubseteq (\forall \mathbf{X}. \text{instEv}(\mathbf{G}_{21}, \mathbf{X}, \mathbf{F}))[\forall \mathbf{X}. \mathbf{c}_1]$$

By the induction hypothesis on $\mathbf{G}_{11} \leq \mathbf{G}_{21}$, we get that

- $[\mathbf{c}_1[\mathbf{F}/\mathbf{X}]]\text{instEv}(\mathbf{G}_{11}, \mathbf{X}, \mathbf{F}) \sqsubseteq \text{instEv}(\mathbf{G}_{21}, \mathbf{X}, \mathbf{F})[\mathbf{c}_1]$

Thus, the result follows immediately, by Lemma B.76.

Case $(\mathbf{B} \leq ?_\delta)$. $\mathbf{G}_1 = \mathbf{B}$ and $\mathbf{G}_2 = ?_\delta$. We divide this case in two; if $\mathbf{X} : \mathbf{X} \in \delta$ or $\mathbf{X} : - \notin \delta$.

- $\mathbf{X} : \mathbf{X} \in \delta = \delta_1; \mathbf{X} : \mathbf{X}; \delta_2$. We know that
 - $\varepsilon_1 = \text{instEv}(\mathbf{G}_1, \mathbf{X}, \mathbf{F}) = \text{instEv}(\mathbf{B}, \mathbf{X}, \mathbf{F}) = \text{reflEv}(\mathbf{B}) = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\}$
 - $\varepsilon_2 = \text{instEv}(\mathbf{G}_2, \mathbf{X}, \mathbf{F}) = \text{instEv}(?_\delta, \mathbf{X}, \mathbf{F}) = [(?_{\delta_1; \delta_2}, \text{inj}_?, \text{inj}_?), (\mathbf{F}, \text{inj}_{\mathbf{X}}, \text{inj}_{\mathbf{F}})]$
 - $\mathbf{c} = \text{initPT}(\mathbf{G}_1, \mathbf{G}_2) = \text{initPT}(\mathbf{B}, ?_\delta) = \text{inj}_{\mathbf{B}}$
 - $\mathbf{c}' = \text{initPT}(\mathbf{G}_1[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}, \mathbf{G}_2[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}) = \text{initPT}(\mathbf{B}[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}, ?_\delta[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}) = \text{initPT}(\mathbf{B}, ?_{\delta_1; \delta_2}) = \text{inj}_{\mathbf{B}}$

Therefore, we are required to prove $[c[F/X]]\varepsilon_1 \sqsubseteq \varepsilon_2[c']$. Or what is the same, we are required to prove $[\text{inj}_B]\{(B, B, B)\} \sqsubseteq [(\delta_1, \delta_2, \text{inj}_?, \text{inj}_?), (F, \text{inj}_X, \text{inj}_F)][\text{inj}_B]$, because $c[F/X] = \text{inj}_B[F/X] = \text{inj}_B$. The results follows immediately because $[\text{inj}_B]\{(B, B, B)\} \sqsubseteq [(\delta_1, \delta_2, \text{inj}_?, \text{inj}_?)]\text{inj}_B$. Or what is the same, for all element in ε_1 (there is only one element in ε_1 , (B, B, B)) there exist an element in ε_2 $(\delta_1, \delta_2, \text{inj}_?, \text{inj}_?)$, c_t , c_l and c_r ($c_t = c_l = c_r = \text{inj}_B$) such that

- $c_t; \text{inj}_? = c_l (\text{inj}_B; \text{inj}_? = \text{inj}_B)$
- $B; c[F/X] = c_l (B; \text{inj}_B = \text{inj}_B)$
- $c_t; \text{inj}_? = c_r (\text{inj}_B; \text{inj}_? = \text{inj}_B)$
- $B; c' = c_r (B; \text{inj}_B = \text{inj}_B)$

Thus, the result holds.

- $X : _ \notin \delta$. We know that

- $\varepsilon_1 = \text{instEv}(G_1, X, F) = \text{instEv}(B, X, F) = \text{reflEv}(B) = \{(B, B, B)\}$
- $\varepsilon_2 = \text{instEv}(G_2, X, F) = \text{instEv}(\delta, X, F) = \text{reflEv}(\delta) = [(\delta, \text{inj}_?, \text{inj}_?)]$
- $c = \text{initPT}(G_1, G_2) = \text{initPT}(B, \delta) = \text{inj}_B$
- $c' = \text{initPT}(G_1[F/X] \setminus X, G_2[F/X] \setminus X) = \text{initPT}(B[F/X] \setminus X, \delta[F/X] \setminus X) = \text{initPT}(B, \delta) = \text{inj}_B$

Therefore, we are required to prove $[c[F/X]]\varepsilon_1 \sqsubseteq \varepsilon_2[c']$. Or what is the same, we are required to prove $[\text{inj}_B]\{(B, B, B)\} \sqsubseteq [(\delta, \text{inj}_?, \text{inj}_?)]\text{inj}_B$, because $c[F/X] = \text{inj}_B[F/X] = \text{inj}_B$. Note that the result follows immediately.

Case ($Y \leq \delta$ and $Y : F \in \delta$). $G_1 = Y$ and $G_2 = \delta$. We divide this case in three cases: $Y = X$ and $X : X \in \delta$, $Y \not\prec X$ and $X : X \in \delta$, and $Y \not\prec X$ and $X : _ \notin \delta$. Note that the case $Y = X$ and $Y : _ \notin \delta$ is not possible because by premise, we know that $Y : F \in \delta$.

- $X : X \in \delta = \delta_1; X : X; \delta_2$. We know that

- $\varepsilon_1 = \text{instEv}(G_1, X, F) = \text{instEv}(X, X, F) = \text{reflEv}(F) = [(F, F, F)]$
- $\varepsilon_2 = \text{instEv}(G_2, X, F) = \text{instEv}(\delta, X, F) = [(\delta_1, \delta_2, \text{inj}_?, \text{inj}_?), (F, \text{inj}_X, \text{inj}_F)]$
- $c = \text{initPT}(G_1, G_2) = \text{initPT}(X, \delta) = \text{inj}_X$
- $c' = \text{initPT}(G_1[F/X] \setminus X, G_2[F/X] \setminus X) = \text{initPT}(X[F/X] \setminus X, \delta[F/X] \setminus X) = \text{initPT}(F, \delta_1; \delta_2) = \text{inj}_F$

Therefore, we are required to prove $[c[F/X]]\varepsilon_1 \sqsubseteq \varepsilon_2[c']$. Or what is the same, we are required to prove $[\text{inj}_X][(F, F, F)] \sqsubseteq [(\delta_1, \delta_2, \text{inj}_?, \text{inj}_?), (F, \text{inj}_X, \text{inj}_F)][\text{inj}_F]$, because $c[F/X] = \text{inj}_X[F/X] = \text{inj}_X$. The results follows immediately because $[\text{inj}_X][(F, F, F)] \sqsubseteq [(F, \text{inj}_X, \text{inj}_F)][\text{inj}_F]$. Or what is the same, for all element in ε_1 (there is only one element in ε_1 , (F, F, F)) there exist an element in ε_2 $(F, \text{inj}_X, \text{inj}_F)$, c_t , c_l and c_r ($c_t = F$, $c_l = \text{inj}_X$ and $c_r = \text{inj}_F$) such that

- $c_t; \text{inj}_X = c_l (F; \text{inj}_X = \text{inj}_X)$
- $F; c[F/X] = c_l (F; \text{inj}_X = \text{inj}_X)$
- $c_t; \text{inj}_F = c_r (F; \text{inj}_F = \text{inj}_F)$

$$- F; \mathbf{c}' = \mathbf{c}_r (F; \text{inj}_F = \text{inj}_F)$$

Thus, the result holds.

- $Y \not\rightsquigarrow X$ and $X : X \in \delta = \delta_1; X : X; \delta_2$. We know that

$$\begin{aligned} - \varepsilon_1 &= \text{instEv}(G_1, X, F) = \text{instEv}(Y, X, F) = \text{reflEv}(Y) = [(Y, Y, Y)] \\ - \varepsilon_2 &= \text{instEv}(G_2, X, F) = \text{instEv}(\delta, X, F) = [(\delta_1; \delta_2, \text{inj}_?, \text{inj}_?), (F, \text{inj}_X, \text{inj}_F)] \\ - \mathbf{c} &= \text{initPT}(G_1, G_2) = \text{initPT}(Y, \delta) = \text{inj}_Y \\ - \mathbf{c}' &= \text{initPT}(G_1[F/X] \setminus X, G_2[F/X] \setminus X) = \text{initPT}(Y[F/X] \setminus X, \delta[F/X] \setminus X) = \text{initPT}(Y, \delta_1; \delta_2) = \\ &\quad \text{inj}_Y \end{aligned}$$

Therefore, we are required to prove $[\mathbf{c}[F/X]]\varepsilon_1 \sqsubseteq \varepsilon_2[\mathbf{c}']$. Or what is the same, we are required to prove $[\text{inj}_Y][[(Y, Y, Y)]] \sqsubseteq [(\delta_1; \delta_2, \text{inj}_?, \text{inj}_?), (F, \text{inj}_X, \text{inj}_F)][[\text{inj}_Y]]$, because $\mathbf{c}[F/X] = \text{inj}_Y[F/X] = \text{inj}_Y$. The results follows immediately because $[\text{inj}_Y][[(Y, Y, Y)]] \sqsubseteq [(\delta_1; \delta_2, \text{inj}_?, \text{inj}_?)][[\text{inj}_Y]]$. Or what is the same, for all element in ε_1 (there is only one element in ε_1 , (Y, Y, Y)) there exist an element in ε_2 ($(\delta_1; \delta_2, \text{inj}_?, \text{inj}_?)$), \mathbf{c}_t , \mathbf{c}_l and \mathbf{c}_r ($\mathbf{c}_t = \mathbf{c}_l = \mathbf{c}_r = \text{inj}_Y$) such that

$$\begin{aligned} - \mathbf{c}_t; \text{inj}_? &= \mathbf{c}_l (\text{inj}_Y; \text{inj}_? = \text{inj}_Y) \\ - Y; \mathbf{c}[F/X] &= \mathbf{c}_l (Y; \text{inj}_Y = \text{inj}_Y) \\ - \mathbf{c}_t; \text{inj}_? &= \mathbf{c}_r (\text{inj}_Y; \text{inj}_? = \text{inj}_Y) \\ - Y; \mathbf{c}' &= \mathbf{c}_r (Y; \text{inj}_Y = \text{inj}_Y) \end{aligned}$$

Thus, the result holds.

- $Y \not\rightsquigarrow X$ and $X : _ \notin \delta$. We know that

$$\begin{aligned} - \varepsilon_1 &= \text{instEv}(G_1, X, F) = \text{instEv}(Y, X, F) = \text{reflEv}(Y) = [(Y, Y, Y)] \\ - \varepsilon_2 &= \text{instEv}(G_2, X, F) = \text{instEv}(\delta, X, F) = \text{reflEv}(\delta) = [(\delta, \text{inj}_?, \text{inj}_?)] \\ - \mathbf{c} &= \text{initPT}(G_1, G_2) = \text{initPT}(Y, \delta) = \text{inj}_Y \\ - \mathbf{c}' &= \text{initPT}(G_1[F/X] \setminus X, G_2[F/X] \setminus X) = \text{initPT}(Y[F/X] \setminus X, \delta[F/X] \setminus X) = \text{initPT}(Y, \delta) = \\ &\quad \text{inj}_Y \end{aligned}$$

Therefore, we are required to prove $[\mathbf{c}[F/X]]\varepsilon_1 \sqsubseteq \varepsilon_2[\mathbf{c}']$. Or what is the same, we are required to prove $[\text{inj}_Y][[(Y, Y, Y)]] \sqsubseteq [(\delta, \text{inj}_?, \text{inj}_?)][[\text{inj}_Y]]$, because $\mathbf{c}[F/X] = \text{inj}_Y[F/X] = \text{inj}_Y$. Note that the result follows immediately.

Case $(\delta \leq \delta'$ and $\delta \subseteq \delta')$. $G_1 = \delta$ and $G_2 = \delta'$. We divide this case in three cases: $X : X \in \delta \subseteq \delta'$, $X : X \notin \delta$ and $X : X \in \delta'$, and $X : X \notin \delta$ and $X : X \notin \delta'$.

- $X : X \in \delta \subseteq \delta'$, $\delta = \delta_1; X : X; \delta_2$ and $\delta' = \delta'_1; X : X; \delta'_2$. We know that

$$\begin{aligned} - \varepsilon_1 &= \text{instEv}(G_1, X, F) = \text{instEv}(\delta, X, F) = [(\delta_1; \delta_2, \text{inj}_?, \text{inj}_?), (F, \text{inj}_X, \text{inj}_F)] \\ - \varepsilon_2 &= \text{instEv}(G_2, X, F) = \text{instEv}(\delta', X, F) = [(\delta'_1; \delta'_2, \text{inj}_?, \text{inj}_?), (F, \text{inj}_X, \text{inj}_F)] \\ - \mathbf{c} &= \text{initPT}(G_1, G_2) = \text{initPT}(\delta, \delta') = \text{inj}_? \\ - \mathbf{c}' &= \text{initPT}(G_1[F/X] \setminus X, G_2[F/X] \setminus X) = \text{initPT}(\delta[F/X] \setminus X, \delta'[F/X] \setminus X) = \\ &\quad \text{initPT}(\delta_1; \delta_2, \delta'_1; \delta'_2) = \text{inj}_? \end{aligned}$$

Therefore, we are required to prove $[c[\mathbf{F}/\mathbf{X}]]\varepsilon_1 \sqsubseteq \varepsilon_2[c']$. Or what is the same, we are required to prove $[\text{inj}_?][(\delta_1, \delta_2, \text{inj}_?, \text{inj}_?), (\mathbf{F}, \text{inj}_\mathbf{X}, \text{inj}_\mathbf{F})] \sqsubseteq [(\delta'_1, \delta'_2, \text{inj}_?, \text{inj}_?), (\mathbf{F}, \text{inj}_\mathbf{X}, \text{inj}_\mathbf{F})][\text{inj}_?]$, because $c[\mathbf{F}/\mathbf{X}] = \text{inj}_?[\mathbf{F}/\mathbf{X}] = \text{inj}_?$. First, we prove that for $(\delta_1, \delta_2, \text{inj}_?, \text{inj}_?) \in \varepsilon_1$ there exist an element in ε_2 $(\delta'_1, \delta'_2, \text{inj}_?, \text{inj}_?)$, c_t , c_l and c_r ($c_t = c_l = c_r = \text{inj}_?$) such that

- $c_t; \text{inj}_? = c_l$ ($\text{inj}_?; \text{inj}_? = \text{inj}_?$)
- $\text{inj}_?; c[\mathbf{F}/\mathbf{X}] = c_l$ ($\text{inj}_?; \text{inj}_? = \text{inj}_?$)
- $c_t; \text{inj}_? = c_r$ ($\text{inj}_?; \text{inj}_? = \text{inj}_?$)
- $\text{inj}_?; c' = c_r$ ($\text{inj}_?; \text{inj}_? = \text{inj}_?$)

Second, we prove that for $(\mathbf{F}, \text{inj}_\mathbf{X}, \text{inj}_\mathbf{F}) \in \varepsilon_1$ there exist an element in ε_2 $(\mathbf{F}, \text{inj}_\mathbf{X}, \text{inj}_\mathbf{F})$, c_t , c_l and c_r ($c_t = \mathbf{F}$, $c_l = \text{inj}_\mathbf{X}$ and $c_r = \text{inj}_\mathbf{F}$) such that

- $c_t; \text{inj}_\mathbf{X} = c_l$ ($\mathbf{F}; \text{inj}_\mathbf{X} = \text{inj}_\mathbf{X}$)
- $\text{inj}_\mathbf{X}; c[\mathbf{F}/\mathbf{X}] = c_l$ ($\text{inj}_\mathbf{X}; \text{inj}_? = \text{inj}_\mathbf{X}$)
- $c_t; \text{inj}_\mathbf{F} = c_r$ ($\mathbf{F}; \text{inj}_\mathbf{F} = \text{inj}_\mathbf{F}$)
- $\text{inj}_\mathbf{F}; c' = c_r$ ($\text{inj}_\mathbf{F}; \text{inj}_? = \text{inj}_\mathbf{F}$)

Thus, the result holds.

- $X : X \notin \delta$ and $X : X \in \delta' = ?_{\delta'_1; X; X\delta'_2}$. We know that

- $\varepsilon_1 = \text{instEv}(\mathbf{G}_1, X, \mathbf{F}) = \text{instEv}(\delta, X, \mathbf{F}) = \text{reflEv}(\delta) = [(\delta, \text{inj}_?, \text{inj}_?)]$
- $\varepsilon_2 = \text{instEv}(\mathbf{G}_2, X, \mathbf{F}) = \text{instEv}(\delta', X, \mathbf{F}) = [(\delta'_1, \delta'_2, \text{inj}_?, \text{inj}_?), (\mathbf{F}, \text{inj}_\mathbf{X}, \text{inj}_\mathbf{F})]$
- $c = \text{initPT}(\mathbf{G}_1, \mathbf{G}_2) = \text{initPT}(\delta, \delta') = \text{inj}_?$
- $c' = \text{initPT}(\mathbf{G}_1[\mathbf{F}/\mathbf{X}] \setminus X, \mathbf{G}_2[\mathbf{F}/\mathbf{X}] \setminus X) = \text{initPT}(\delta[\mathbf{F}/\mathbf{X}] \setminus X, \delta'[\mathbf{F}/\mathbf{X}] \setminus X) = \text{initPT}(\delta, \delta') = \text{inj}_?$

Therefore, we are required to prove $[c[\mathbf{F}/\mathbf{X}]]\varepsilon_1 \sqsubseteq \varepsilon_2[c']$. Or what is the same, we are required to prove $[\text{inj}_?][(\delta, \text{inj}_?, \text{inj}_?) \sqsubseteq [(\delta'_1, \delta'_2, \text{inj}_?, \text{inj}_?), (\mathbf{F}, \text{inj}_\mathbf{X}, \text{inj}_\mathbf{F})][\text{inj}_?]$, because $c[\mathbf{F}/\mathbf{X}] = \text{inj}_?[\mathbf{F}/\mathbf{X}] = \text{inj}_?$. The results follows immediately because $[\text{inj}_?][(\delta, \text{inj}_?, \text{inj}_?) \sqsubseteq [(\delta'_1, \delta'_2, \text{inj}_?, \text{inj}_?)][\text{inj}_?]$. Or what is the same, for all element in ε_1 (there is only one element in ε_1 , $(\delta, \text{inj}_?, \text{inj}_?)$) there exist an element in ε_2 $(\delta'_1, \delta'_2, \text{inj}_?, \text{inj}_?)$, c_t , c_l and c_r ($c_t = c_l = c_r = \text{inj}_?$) such that

- $c_t; \text{inj}_? = c_l$ ($\text{inj}_?; \text{inj}_? = \text{inj}_?$)
- $\text{inj}_?; c[\mathbf{F}/\mathbf{X}] = c_l$ ($\text{inj}_?; \text{inj}_? = \text{inj}_?$)
- $c_t; \text{inj}_? = c_r$ ($\text{inj}_?; \text{inj}_? = \text{inj}_?$)
- $\text{inj}_?; c' = c_r$ ($\text{inj}_?; \text{inj}_? = \text{inj}_?$)

Thus, the result holds.

- $X : X \notin \delta$ and $X : X \notin \delta'$. We know that

- $\varepsilon_1 = \text{instEv}(\mathbf{G}_1, X, \mathbf{F}) = \text{instEv}(\delta, X, \mathbf{F}) = \text{reflEv}(\delta) = [(\delta, \text{inj}_?, \text{inj}_?)]$
- $\varepsilon_2 = \text{instEv}(\mathbf{G}_2, X, \mathbf{F}) = \text{instEv}(\delta', X, \mathbf{F}) = \text{reflEv}(\delta') = [(\delta', \text{inj}_?, \text{inj}_?)]$
- $c = \text{initPT}(\mathbf{G}_1, \mathbf{G}_2) = \text{initPT}(\delta, \delta') = \text{inj}_?$

$$- \mathbf{c}' = \text{initPT}(\mathbf{G}_1[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}, \mathbf{G}_2[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}) = \text{initPT}(\mathbf{?}_\delta[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}, \mathbf{?}_{\delta'}[\mathbf{F}/\mathbf{X}] \setminus \mathbf{X}) = \text{initPT}(\mathbf{?}_\delta, \mathbf{?}_{\delta'}) = \mathbf{inj}_?$$

Therefore, we are required to prove $[\mathbf{c}[\mathbf{F}/\mathbf{X}]]_{\varepsilon_1} \sqsubseteq \varepsilon_2[\mathbf{c}']$. Or what is the same, we are required to prove $[\mathbf{inj}_?][(\mathbf{?}_\delta, \mathbf{inj}_?, \mathbf{inj}_?)] \sqsubseteq [(\mathbf{?}_{\delta'}, \mathbf{inj}_?, \mathbf{inj}_?)][\mathbf{inj}_?]$, because $\mathbf{c}[\mathbf{F}/\mathbf{X}] = \mathbf{inj}_?[\mathbf{F}/\mathbf{X}] = \mathbf{inj}_?$. Note that the result follows immediately. □

Lemma B.126 *If $(\Gamma_1, \Gamma_2) \equiv \Omega$ then $\Gamma_1 \sqsubseteq \Gamma_2$.*

PROOF. Directly by Definition B.109 and $(\Gamma_1, \Gamma_2) \equiv \Omega$. □

Lemma B.127 *If $\mathbf{G}_1 \leq \mathbf{G}_2$ then $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$.*

PROOF. We proceed by induction on $\mathbf{G}_1 \leq \mathbf{G}_2$.

Case $(\mathbf{B} \leq \mathbf{B})$. The results follows immediately by the rule $\mathbf{B} \sqsubseteq \mathbf{B}$.

Case $(\mathbf{X} \leq \mathbf{X})$. The results follows immediately by the rule $\mathbf{X} \sqsubseteq \mathbf{X}$.

Case $(\mathbf{B} \leq \mathbf{?}_\delta)$. The results follows immediately by the rule $\mathbf{B} \sqsubseteq \mathbf{?}_\delta$.

Case $(\mathbf{X} \leq \mathbf{?}_\delta$ and $\mathbf{X} : \mathbf{X} \in \delta)$. The results follows immediately by the rule $\mathbf{X} \sqsubseteq \mathbf{?}_\delta$.

Case $(\mathbf{G}_{11} \rightarrow \mathbf{G}_{12} \leq \mathbf{G}_{21} \rightarrow \mathbf{G}_{22})$. We know that

$$\frac{\mathbf{G}_{11} \leq \mathbf{G}_{21} \quad \mathbf{G}_{12} \leq \mathbf{G}_{22}}{\mathbf{G}_{11} \rightarrow \mathbf{G}_{12} \leq \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}}$$

By the induction hypothesis on $\mathbf{G}_{11} \leq \mathbf{G}_{21}$ and $\mathbf{G}_{12} \leq \mathbf{G}_{22}$, we know that $\mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$. We are required to prove that $\mathbf{G}_{11} \rightarrow \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}$, which follows immediately by the rule

$$\frac{\mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21} \quad \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}}{\mathbf{G}_{11} \rightarrow \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{21} \rightarrow \mathbf{G}_{22}}$$

Case $(\mathbf{G}_{11} \times \mathbf{G}_{12} \leq \mathbf{G}_{21} \times \mathbf{G}_{22})$. We know that

$$\frac{\mathbf{G}_{11} \leq \mathbf{G}_{21} \quad \mathbf{G}_{12} \leq \mathbf{G}_{22}}{\mathbf{G}_{11} \times \mathbf{G}_{12} \leq \mathbf{G}_{21} \times \mathbf{G}_{22}}$$

By the induction hypothesis on $\mathbf{G}_{11} \leq \mathbf{G}_{21}$ and $\mathbf{G}_{12} \leq \mathbf{G}_{22}$, we know that $\mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21}$ and $\mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}$. We are required to prove that $\mathbf{G}_{11} \times \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{21} \times \mathbf{G}_{22}$, which follows immediately by the rule

$$\frac{\mathbf{G}_{11} \sqsubseteq \mathbf{G}_{21} \quad \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{22}}{\mathbf{G}_{11} \times \mathbf{G}_{12} \sqsubseteq \mathbf{G}_{21} \times \mathbf{G}_{22}}$$

Case $(\forall \mathbf{X}. \mathbf{G}_{11} \leq \forall \mathbf{X}. \mathbf{G}_{12})$. We know that

$$\frac{\mathbf{G}_{11} \leq \mathbf{G}_{12}}{\forall \mathbf{X}. \mathbf{G}_{11} \leq \forall \mathbf{X}. \mathbf{G}_{12}}$$

By the induction hypothesis on $\mathbf{G}_{11} \leq \mathbf{G}_{12}$, we know that $\mathbf{G}_{11} \sqsubseteq \mathbf{G}_{12}$. We are required to prove that $\forall \mathbf{X}. \mathbf{G}_{11} \sqsubseteq \forall \mathbf{X}. \mathbf{G}_{12}$, which follows immediately by the rule

$$\frac{\mathbf{G}_{11} \sqsubseteq \mathbf{G}_{12}}{\forall \mathbf{X}. \mathbf{G}_{11} \sqsubseteq \forall \mathbf{X}. \mathbf{G}_{12}}$$

Case $(?_{\delta} \leq ?_{\delta'} \text{ and } \delta \subseteq \delta')$. The results follows immediately by the rule $?_{\delta} \sqsubseteq ?_{\delta'}$.

□

Lemma B.128 *If $G_1 = G_2$, then $G_1 \sqsubseteq G_2$.*

PROOF. Since, $G_1 = G_2$, we have that $\gamma(G_1) = \gamma(G_2)$, and therefore, $\gamma(G_1) \subseteq \gamma(G_2)$. Thus, by Definition B.94, we have that $G_1 \sqsubseteq G_2$, and the result holds. □

Lemma B.129 *If $G \rightarrow G'$, then $G' \sqsubseteq G$.*

PROOF. We follow by case analysis on $G \rightarrow G'$.

Case $(G_1 \rightarrow G_2 \rightarrow G_1 \rightarrow G_2)$. Immediately, by Lemma B.128.

Case $(\forall X.G \rightarrow \forall X.G)$. Immediately, by Lemma B.128.

Case $(G_1 \times G_2 \rightarrow G_1 \times G_2)$. Immediately, by Lemma B.128.

Case $(?_{\delta} \rightarrow ?_{\delta} \rightarrow ?_{\delta})$. We know that $\text{ftv}(?_{\delta} \rightarrow ?_{\delta}) = \text{ftv}(?_{\delta})$, and therefore, $\text{ftv}(?_{\delta} \rightarrow ?_{\delta}) \subseteq \text{ftv}(?_{\delta})$. Thus, by Lemma B.100, the result hold.

Case $(?_{\delta} \rightarrow \forall X.?_{\delta, X:X})$. We know that $\text{ftv}(\forall X.?_{\delta, X:X}) = \text{ftv}(?_{\delta})$, and therefore, $\text{ftv}(\forall X.?_{\delta, X:X}) \subseteq \text{ftv}(?_{\delta})$. Thus, by Lemma B.100, the result hold.

Case $(?_{\delta} \rightarrow ?_{\delta} \times ?_{\delta})$. We know that $\text{ftv}(?_{\delta} \times ?_{\delta}) = \text{ftv}(?_{\delta})$, and therefore, $\text{ftv}(?_{\delta} \times ?_{\delta}) \subseteq \text{ftv}(?_{\delta})$. Thus, by Lemma B.100, the result hold.

□

Lemma B.130 *If $G_1 \sim G_2$, $G_1 \sqsubseteq G'_1$ and $G_2 \sqsubseteq G'_2$, then $G'_1 \sim G'_2$.*

PROOF. Since $G_1 \sqsubseteq G'_1$ and $G_2 \sqsubseteq G'_2$, by Definition B.94, we get that $\gamma(G_1) \subseteq \gamma(G'_1)$ and $\gamma(G_2) \subseteq \gamma(G'_2)$. Since $G_1 \sim G_2$, by Definition B.96, we know that there, exist T_1 and T_2 , such that $T_1 = T_2$, $T_1 \in \gamma(G_1)$ and $T_2 \in \gamma(G_2)$. Therefore, we have that $T_1 \in \gamma(G_1) \subseteq \gamma(G'_1)$ and $T_2 \in \gamma(G_2) \subseteq \gamma(G'_2)$, and by Definition B.96, the result holds ($G'_1 \sim G'_2$). □

Lemma B.131 *If $G_1 \sqsubseteq G_2$, $G_1 \rightarrow G_{11} \rightarrow G_{12}$ and $G_2 \rightarrow G_{21} \rightarrow G_{22}$, then $G_{11} \rightarrow G_{12} \sqsubseteq G_{21} \rightarrow G_{22}$.*

PROOF. By definition of $G_1 \rightarrow G_{11} \rightarrow G_{12}$ and $G_2 \rightarrow G_{21} \rightarrow G_{22}$, we get the following cases.

- $G_1 = G_{11} \rightarrow G_{12}$ and $G_2 = G_{21} \rightarrow G_{22}$. The result holds, since $G_1 \sqsubseteq G_2$.
- $G_1 = G_{11} \rightarrow G_{12}$, $G_2 = ?_{\delta}$ and $G_2 \rightarrow ?_{\delta} \rightarrow ?_{\delta}$. We are required to show that $G_{11} \rightarrow G_{12} \sqsubseteq ?_{\delta} \rightarrow ?_{\delta}$, which follows immediately since $G_{11} \rightarrow G_{12} \sqsubseteq ?_{\delta}$.
- $G_1 = ?_{\delta}$ and $G_2 = G_{21} \rightarrow G_{22}$. This case is not possible since is not the case that $?_{\delta} \sqsubseteq G_{11} \rightarrow G_{12}$.
- $G_1 = ?_{\delta}$, $G_2 = ?_{\delta'}$, $G_1 \rightarrow ?_{\delta} \rightarrow ?_{\delta}$ and $G_2 \rightarrow ?_{\delta'} \rightarrow ?_{\delta'}$. We are required to prove that $?_{\delta} \rightarrow ?_{\delta} \sqsubseteq ?_{\delta'} \rightarrow ?_{\delta'}$, which follows immediately since $?_{\delta} \sqsubseteq ?_{\delta'}$.

□

Lemma B.132 *If $G_1 \sqsubseteq G_2$, $G_1 \rightarrow G_{11} \times G_{12}$ and $G_2 \rightarrow G_{21} \times G_{22}$, then $G_{11} \times G_{12} \sqsubseteq G_{21} \times G_{22}$.*

PROOF. By definition of $G_1 \rightarrow G_{11} \times G_{12}$ and $G_2 \rightarrow G_{21} \times G_{22}$, we get the following cases.

- $G_1 = G_{11} \times G_{12}$ and $G_2 = G_{21} \times G_{22}$. The result holds, since $G_1 \sqsubseteq G_2$.
- $G_1 = G_{11} \times G_{12}$, $G_2 = ?_\delta$ and $G_2 \rightarrow ?_\delta \times ?_\delta$. We are required to show that $G_{11} \times G_{12} \sqsubseteq ?_\delta \times ?_\delta$, which follows immediately since $G_{11} \times G_{12} \sqsubseteq ?_\delta$.
- $G_1 = ?_\delta$ and $G_2 = G_{21} \times G_{22}$. This case is not possible since is not the case that $?_\delta \sqsubseteq G_{11} \times G_{12}$.
- $G_1 = ?_\delta$, $G_2 = ?_{\delta'}$, $G_1 \rightarrow ?_\delta \times ?_\delta$ and $G_2 \rightarrow ?_{\delta'} \times ?_{\delta'}$. We are required to prove that $?_\delta \times ?_\delta \sqsubseteq ?_{\delta'} \times ?_{\delta'}$, which follows immediately since $?_\delta \sqsubseteq ?_{\delta'}$.

□

Lemma B.133 *If $G_1 \sqsubseteq G_2$, $G_1 \rightarrow \forall X.G_{11}$ and $G_2 \rightarrow \forall X.G_{21}$, then $\forall X.G_{11} \sqsubseteq \forall X.G_{21}$.*

PROOF. By definition of $G_1 \rightarrow \forall X.G_{11}$ and $G_2 \rightarrow \forall X.G_{21}$, we get the following cases.

- $G_1 = \forall X.G_{11}$ and $G_2 = \forall X.G_{21}$. The result holds, since $G_1 \sqsubseteq G_2$.
- $G_1 = \forall X.G_{11}$, $G_2 = ?_\delta$ and $G_2 \rightarrow \forall X.?_{\delta,X:X}$. We are required to show that $\forall X.G_{11} \sqsubseteq \forall X.?_{\delta,X:X}$, which follows immediately since $\forall X.G_{11} \sqsubseteq ?_\delta$.
- $G_1 = ?_\delta$ and $G_2 = \forall X.G_{21}$. This case is not possible since is not the case that $?_\delta \sqsubseteq \forall X.G_{11}$.
- $G_1 = ?_\delta$, $G_2 = ?_{\delta'}$, $G_1 \rightarrow \forall X.?_{\delta,X:X}$ and $G_2 \rightarrow \forall X.?_{\delta',X:X}$. We are required to prove that $\forall X.?_{\delta,X:X} \sqsubseteq \forall X.?_{\delta',X:X}$, which follows immediately since $?_\delta \sqsubseteq ?_{\delta'}$, and therefore, $?_{\delta,X:X} \sqsubseteq ?_{\delta',X:X}$.

□

Lemma B.134 $G_1 \rightarrow G'_1 \sqsubseteq G_2 \rightarrow G'_2$ if and only if $dom^\sharp(G_1 \rightarrow G'_1) \sqsubseteq dom^\sharp(G_2 \rightarrow G'_2)$ and $cod^\sharp(G_1 \rightarrow G'_1) \sqsubseteq cod^\sharp(G_2 \rightarrow G'_2)$.

PROOF. We know that $dom^\sharp(G_1 \rightarrow G'_1) = G_1$, $dom^\sharp(G_2 \rightarrow G'_2) = G_2$, $cod^\sharp(G_1 \rightarrow G'_1) = G'_1$ and $cod^\sharp(G_2 \rightarrow G'_2) = G'_2$. We know by $(\sqsubseteq \rightarrow)$ rule, that $G_1 \rightarrow G'_1 \sqsubseteq G_2 \rightarrow G'_2$ if and only if $G_1 \sqsubseteq G_2$ and $G'_1 \sqsubseteq G'_2$. Therefore, the result holds. □

Lemma B.135 $G_1 \times G'_1 \sqsubseteq G_2 \times G'_2$ if and only if $proj_1^\sharp(G_1 \times G'_1) \sqsubseteq proj_1^\sharp(G_2 \times G'_2)$ and $proj_2^\sharp(G_1 \times G'_1) \sqsubseteq proj_2^\sharp(G_2 \times G'_2)$.

PROOF. We know that $proj_1^\sharp(G_1 \times G'_1) = G_1$, $proj_1^\sharp(G_2 \times G'_2) = G_2$, $proj_2^\sharp(G_1 \times G'_1) = G'_1$ and $proj_2^\sharp(G_2 \times G'_2) = G'_2$. We know by $(\sqsubseteq \times)$ rule, that $G_1 \times G'_1 \sqsubseteq G_2 \times G'_2$ if and only if $G_1 \sqsubseteq G_2$ and $G'_1 \sqsubseteq G'_2$. Therefore, the result holds. □

Lemma B.136 $\mathbf{G} \rightarrow \mathbf{G}_1 \rightarrow \mathbf{G}_2$ if and only if $\text{dom}^\sharp(\mathbf{G}) = \mathbf{G}_1$ and $\text{cod}^\sharp(\mathbf{G}) = \mathbf{G}_2$.

PROOF. If $\mathbf{G} \rightarrow \mathbf{G}_1 \rightarrow \mathbf{G}_2$, we know that $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$, and therefore, $\text{dom}^\sharp(\mathbf{G}) = \mathbf{G}_1$ and $\text{cod}^\sharp(\mathbf{G}) = \mathbf{G}_2$. Thus, the result holds. If $\text{dom}^\sharp(\mathbf{G}) = \mathbf{G}_1$ and $\text{cod}^\sharp(\mathbf{G}) = \mathbf{G}_2$, we know that $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$, or $\mathbf{G} = ?_\delta$, $\mathbf{G}_1 = ?_\delta$ and $\mathbf{G}_2 = ?_\delta$. If $\mathbf{G} = \mathbf{G}_1 \rightarrow \mathbf{G}_2$, we know that $\mathbf{G} \rightarrow \mathbf{G}_1 \rightarrow \mathbf{G}_2$, and the result hold. If $\mathbf{G} = ?_\delta$, then $\mathbf{G} \rightarrow ?_\delta \rightarrow ?_\delta$ and the result holds. \square

Lemma B.137 $\mathbf{G} \rightarrow \mathbf{G}_1 \times \mathbf{G}_2$ if and only if $\text{proj}_i^\sharp(\mathbf{G}) = \mathbf{G}_i$.

PROOF. If $\mathbf{G} \rightarrow \mathbf{G}_1 \times \mathbf{G}_2$, we know that $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$, and therefore, $\text{proj}_1^\sharp(\mathbf{G}) = \mathbf{G}_1$ and $\text{proj}_2^\sharp(\mathbf{G}) = \mathbf{G}_2$. Thus, the result holds. If $\text{proj}_i^\sharp(\mathbf{G}) = \mathbf{G}_i$ and $\text{proj}_i^\sharp(\mathbf{G}) = \mathbf{G}_2$, we know that $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$, or $\mathbf{G} = ?_\delta$, $\mathbf{G}_1 = ?_\delta$ and $\mathbf{G}_2 = ?_\delta$. If $\mathbf{G} = \mathbf{G}_1 \times \mathbf{G}_2$, we know that $\mathbf{G} \rightarrow \mathbf{G}_1 \times \mathbf{G}_2$, and the result hold. If $\mathbf{G} = ?_\delta$, then $\mathbf{G} \rightarrow ?_\delta \times ?_\delta$ and the result holds. \square

Lemma B.138 $\mathbf{G} \rightarrow \forall X. \mathbf{G}_1$ if and only if $\text{scheme}^\sharp(\mathbf{G}) = \mathbf{G}_1$ and $\mathbf{G}_1[\mathbf{G}'/X] \setminus X = \text{inst}^\sharp(\mathbf{G}, \mathbf{G}')$.

PROOF. If $\mathbf{G} \rightarrow \forall X. \mathbf{G}_1$, we know that $\mathbf{G} = \forall X. \mathbf{G}_1$, and therefore, $\text{scheme}^\sharp(\mathbf{G}) = \mathbf{G}_1$, and therefore, $\mathbf{G}_1[\mathbf{G}'/X] \setminus X = \text{inst}^\sharp(\mathbf{G}, \mathbf{G}') = \text{inst}^\sharp(\forall X. \mathbf{G}_1, \mathbf{G}')$. Thus, the result holds. If $\text{scheme}^\sharp(\mathbf{G}) = \mathbf{G}_1$ and $\mathbf{G}_1[\mathbf{G}'/X] \setminus X = \text{inst}^\sharp(\mathbf{G}, \mathbf{G}')$, we know that $\mathbf{G} = \forall X. \mathbf{G}_1$, or $\mathbf{G} = ?_\delta$, $\mathbf{G}_1 = ?_{\delta, X: X}$. If $\mathbf{G} = \forall X. \mathbf{G}_1$, we know that $\mathbf{G} \rightarrow \forall X. \mathbf{G}_1$, and the result hold. If $\mathbf{G} = ?_\delta$, then

$\mathbf{G} \rightarrow \forall X. ?_{\delta, X: X}$ and the result holds. \square

Lemma B.139 If $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$ and $\text{dom}^\sharp(\mathbf{G}_1)$ is defined, then $\text{dom}^\sharp(\mathbf{G}_1) \sqsubseteq \text{dom}^\sharp(\mathbf{G}_2)$ and $\text{cod}^\sharp(\mathbf{G}_1) \sqsubseteq \text{cod}^\sharp(\mathbf{G}_2)$.

PROOF. Since $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, we know by Definition B.94, that $\gamma(\mathbf{G}_1) \subseteq \gamma(\mathbf{G}_2)$. Since $\text{dom}^\sharp(\mathbf{G}_1)$ is defined, we know that $\text{dom}^\sharp(\mathbf{G}_1) = \alpha(\{ \text{dom}(T) \mid T \in \gamma(\mathbf{G}_1) \})$. Therefore, we get that $\text{dom}^\sharp(\mathbf{G}_2) = \alpha(\{ \text{dom}(T) \mid T \in \gamma(\mathbf{G}_2) \})$ is defined and $A = \{ \text{dom}(T) \mid T \in \gamma(\mathbf{G}_1) \} \subseteq \{ \text{dom}(T) \mid T \in \gamma(\mathbf{G}_2) \} = B$. By Lemma B.92, we get that $A \subseteq \gamma(\alpha(A)) = \gamma(\text{dom}^\sharp(\mathbf{G}_1))$ and $B \subseteq \gamma(\alpha(B)) = \gamma(\text{dom}^\sharp(\mathbf{G}_2))$, where $A \subseteq B \subseteq \gamma(\alpha(B)) = \gamma(\text{dom}^\sharp(\mathbf{G}_2))$. Since $A \subseteq \gamma(\text{dom}^\sharp(\mathbf{G}_2))$, by Lemma B.93, we get that $\text{dom}^\sharp(\mathbf{G}_1) = \alpha(A) \sqsubseteq \text{dom}^\sharp(\mathbf{G}_2)$, as we are required to prove.

Since $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, we know by Definition B.94, that $\gamma(\mathbf{G}_1) \subseteq \gamma(\mathbf{G}_2)$. Since $\text{cod}^\sharp(\mathbf{G}_1)$ is defined, we know that $\text{cod}^\sharp(\mathbf{G}_1) = \alpha(\{ \text{dom}(T) \mid T \in \gamma(\mathbf{G}_1) \})$. Therefore, we get that $\text{cod}^\sharp(\mathbf{G}_2) = \alpha(\{ \text{dom}(T) \mid T \in \gamma(\mathbf{G}_2) \})$ is defined and $A = \{ \text{dom}(T) \mid T \in \gamma(\mathbf{G}_1) \} \subseteq \{ \text{dom}(T) \mid T \in \gamma(\mathbf{G}_2) \} = B$. By Lemma B.92, we get that $A \subseteq \gamma(\alpha(A)) = \gamma(\text{cod}^\sharp(\mathbf{G}_1))$ and $B \subseteq \gamma(\alpha(B)) = \gamma(\text{cod}^\sharp(\mathbf{G}_2))$, where $A \subseteq B \subseteq \gamma(\alpha(B)) = \gamma(\text{cod}^\sharp(\mathbf{G}_2))$. Since $A \subseteq \gamma(\text{cod}^\sharp(\mathbf{G}_2))$, by Lemma B.93, we get that $\text{cod}^\sharp(\mathbf{G}_1) = \alpha(A) \sqsubseteq \text{cod}^\sharp(\mathbf{G}_2)$, as we are required to prove. \square

Lemma B.140 If $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$ and $\text{proj}_i^\sharp(\mathbf{G}_1)$ is defined, then $\text{proj}_i^\sharp(\mathbf{G}_1) \sqsubseteq \text{proj}_i^\sharp(\mathbf{G}_2)$.

PROOF. Since $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, we know by Definition B.94, that $\gamma(\mathbf{G}_1) \subseteq \gamma(\mathbf{G}_2)$. Since $\text{proj}_i^\sharp(\mathbf{G}_1)$ is defined, we know that $\text{proj}_i^\sharp(\mathbf{G}_1) = \alpha(\{ \text{proj}_i(T) \mid T \in \gamma(\mathbf{G}_1) \})$. Therefore, we get that $\text{proj}_i^\sharp(\mathbf{G}_2) = \alpha(\{ \text{proj}_i(T) \mid T \in \gamma(\mathbf{G}_2) \})$ is defined and $A = \{ \text{proj}_i(T) \mid T \in \gamma(\mathbf{G}_1) \} \subseteq \{ \text{proj}_i(T) \mid T \in \gamma(\mathbf{G}_2) \} =$

B . By Lemma B.92, we get that $A \subseteq \gamma(\alpha(A)) = \gamma(\text{proj}_i^\sharp(\mathbf{G}_1))$ and $B \subseteq \gamma(\alpha(B)) = \gamma(\text{proj}_i^\sharp(\mathbf{G}_2))$, where $A \subseteq B \subseteq \gamma(\alpha(B)) = \gamma(\text{proj}_i^\sharp(\mathbf{G}_2))$. Since $A \subseteq \gamma(\text{proj}_i^\sharp(\mathbf{G}_2))$, by Lemma B.93, we get that $\text{proj}_i^\sharp(\mathbf{G}_1) = \alpha(A) \sqsubseteq \text{proj}_i^\sharp(\mathbf{G}_2)$, as we are required to prove. \square

Lemma B.141 *If $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$, $\mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$ and $\text{inst}^\sharp(\mathbf{G}_1, \mathbf{G}'_1)$ is defined, then $\text{inst}^\sharp(\mathbf{G}_1, \mathbf{G}'_1) \sqsubseteq \text{inst}^\sharp(\mathbf{G}_2, \mathbf{G}'_2)$.*

PROOF. Since $\mathbf{G}_1 \sqsubseteq \mathbf{G}_2$ and $\mathbf{G}'_1 \sqsubseteq \mathbf{G}'_2$, we know by Definition B.94, that $\gamma(\mathbf{G}_1) \subseteq \gamma(\mathbf{G}_2)$ and $\gamma(\mathbf{G}'_1) \subseteq \gamma(\mathbf{G}'_2)$. Since $\text{inst}^\sharp(\mathbf{G}_1)$ is defined, we know that $\text{inst}^\sharp(\mathbf{G}_1) = \alpha(\{\text{inst}^\sharp(T, T') \mid T \in \gamma(\mathbf{G}_1) \wedge T' \in \gamma(\mathbf{G}'_1)\})$. Therefore, we get that $\text{inst}^\sharp(\mathbf{G}_2) = \alpha(\{\text{inst}^\sharp(T, T') \mid T \in \gamma(\mathbf{G}_2) \wedge T' \in \gamma(\mathbf{G}'_2)\})$ is defined and $A = \{\text{inst}^\sharp(T, T') \mid T \in \gamma(\mathbf{G}_1) \wedge T' \in \gamma(\mathbf{G}'_1)\} \subseteq \{\text{inst}^\sharp(T, T') \mid T \in \gamma(\mathbf{G}_2) \wedge T' \in \gamma(\mathbf{G}'_2)\} = B$. By Lemma B.92, we get that $A \subseteq \gamma(\alpha(A)) = \gamma(\text{inst}^\sharp(\mathbf{G}_1, \mathbf{G}'_1))$ and $B \subseteq \gamma(\alpha(B)) = \gamma(\text{inst}^\sharp(\mathbf{G}_2, \mathbf{G}'_2))$, where $A \subseteq B \subseteq \gamma(\alpha(B)) = \gamma(\text{inst}^\sharp(\mathbf{G}_2, \mathbf{G}'_2))$. Since $A \subseteq \gamma(\text{inst}^\sharp(\mathbf{G}_2, \mathbf{G}'_2))$, by Lemma B.93, we get that $\text{inst}^\sharp(\mathbf{G}_1, \mathbf{G}'_1) = \alpha(A) \sqsubseteq \text{inst}^\sharp(\mathbf{G}_2, \mathbf{G}'_2)$, as we are required to prove. \square

Lemma B.142 *If $\vdash \mathbf{t}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{t}_2 : \mathbf{G}_2$, $\vdash \mathbf{t}_1 : \mathbf{G}_1 \rightsquigarrow \mathbf{t}_1$ and $\vdash \mathbf{t}_2 : \mathbf{G}_2 \rightsquigarrow \mathbf{t}_2$, then $\vdash \text{initPT}(\mathbf{G}_1, \mathbf{G}_2) : \mathbf{t}_1 \sqsubseteq \mathbf{t}_2$.*

PROOF. Straightforward, by Lemma B.143. \square

Lemma B.143 *If*

- $\Omega \vdash \mathbf{t}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{t}_2 : \mathbf{G}_2$
- $\Delta; \Gamma_1 \vdash \mathbf{t}_1 : \mathbf{G}_1$
- $(\Gamma_1, \Gamma_2) \equiv \Omega$
- $\Delta; \Gamma_1 \vdash \mathbf{t}_1 : \mathbf{G}_1 \rightsquigarrow \mathbf{t}'_1$
- $\Delta; \Gamma_2 \vdash \mathbf{t}_2 : \mathbf{G}_2 \rightsquigarrow \mathbf{t}'_2$
- $\mathbf{c} = \text{initPT}(\mathbf{G}_1, \mathbf{G}_2)$
- $\Omega \rightsquigarrow \Omega$

then $\Omega \vdash \mathbf{c} : \mathbf{t}'_1 \sqsubseteq \mathbf{t}'_2$.

PROOF. We proceed by induction on $\Omega \vdash \mathbf{t}_1 : \mathbf{G}_1 \sqsubseteq \mathbf{t}_2 : \mathbf{G}_2$.

Case (Base values). $\mathbf{t}_1 = \mathbf{b}$ and $\mathbf{t}_2 = \mathbf{b}$. Thus, we know that

$$\frac{}{\Omega \vdash \mathbf{b} : \mathbf{B} \sqsubseteq \mathbf{b} : \mathbf{B}} \quad (\text{Eb}) \frac{\theta(\mathbf{b}) = \mathbf{B}}{\Delta; \Gamma_1 \vdash \mathbf{b} : \mathbf{B} \rightsquigarrow \text{reflEv}(\mathbf{B}) \mathbf{b} :: \mathbf{B}}$$

$$(\text{Eb}) \frac{\theta(\mathbf{b}) = \mathbf{B}}{\Delta; \Gamma_2 \vdash \mathbf{b} : \mathbf{B} \rightsquigarrow \text{reflEv}(\mathbf{B}) \mathbf{b} :: \mathbf{B}}$$

Therefore, we are required to prove that $\Omega \vdash \mathbf{c} : (\text{reflEv}(\mathbf{B}) \mathbf{b} :: \mathbf{B}) \sqsubseteq (\text{reflEv}(\mathbf{B}) \mathbf{b} :: \mathbf{B})$, where $\mathbf{c} = \text{initPT}(\mathbf{B}, \mathbf{B}) = \text{refl}_{\mathbf{B}}$ and $\text{reflEv}(\mathbf{B}) = (\mathbf{B}, \text{refl}_{\mathbf{B}}, \text{refl}_{\mathbf{B}})$. Or what is the same, by Rule ($\sqsubseteq x$), we are required to prove that $[\mathbf{B}] \text{reflEv}(\mathbf{B}) \sqsubseteq \text{reflEv}(\mathbf{B})[\mathbf{c}]$ and $\mathbf{c} : \mathbf{B} \sqsubseteq \mathbf{B}$, which follows immediately. Thus, the result holds.

Case (Variables). $t_1 = x$ and $t_2 = x$.

We know that

$$\frac{x : G_1 \sqsubseteq G_2 \in \Omega}{\Omega \vdash x : G_1 \sqsubseteq x : G_2} \quad (\text{Ex}) \frac{x : G_1 \in \Gamma_1 \quad \Delta \vdash \Gamma_1}{\Delta; \Gamma_1 \vdash x : G_1 \rightsquigarrow x} \quad (\text{Ex}) \frac{x : G_2 \in \Gamma_2 \quad \Delta \vdash \Gamma_2}{\Delta; \Gamma_2 \vdash x : G_2 \rightsquigarrow x}$$

We are required to prove that $\Omega \vdash c : x \sqsubseteq x$, where $c = \text{initPT}(G_1, G_2)$. Or what is the same, we are required to prove that $x \mapsto c : G_1 \sqsubseteq G_2 \in \Omega$, which follows immediately by the definition of $\Omega \rightsquigarrow \Omega$. Thus, the result holds.

Case (Functions). $t_1 = \lambda x : G_3.t_3$ and $t_2 = \lambda x : G_4.t_4$.

We know that

$$\frac{\Omega, x : G_3 \sqsubseteq G_4 \vdash t_3 : G'_3 \sqsubseteq t_4 : G'_4 \quad G_3 \sqsubseteq G_4}{\Omega \vdash \lambda x : G_3.t_3 : G_3- \longrightarrow G'_3 \sqsubseteq \lambda x : G_4.t_4 : G_4- \longrightarrow G'_4}$$

$$(\text{E}\lambda) \frac{\Delta; \Gamma_1, x : G_3 \vdash t_3 : G'_3 \rightsquigarrow t'_3}{\Delta; \Gamma_1 \vdash \lambda x : G_3.t_3 : G_3 \rightarrow G'_3 \rightsquigarrow \text{reflEv}(G_3 \rightarrow G'_3) (\lambda x : G_3.t'_3) :: G_3 \rightarrow G'_3}$$

$$(\text{E}\lambda) \frac{\Delta; \Gamma_2, x : G_4 \vdash t_4 : G'_4 \rightsquigarrow t'_4}{\Delta; \Gamma_2 \vdash \lambda x : G_4.t_4 : G_4 \rightarrow G'_4 \rightsquigarrow \text{reflEv}(G_4 \rightarrow G'_4) (\lambda x : G_4.t'_4) :: G_4 \rightarrow G'_4}$$

We are required to prove that

$$\Omega \vdash c : \text{reflEv}(G_3 \rightarrow G'_3) (\lambda x : G_3.t'_3) :: G_3 \rightarrow G'_3 \sqsubseteq \text{reflEv}(G_4 \rightarrow G'_4) (\lambda x : G_4.t'_4) :: G_4 \rightarrow G'_4$$

where $c = \text{initPT}(G_3 \rightarrow G'_3, G_4 \rightarrow G'_4) = \text{initPT}(G_3, G_4) \longrightarrow \text{initPT}(G'_3, G'_4)$.

We know, by the induction hypothesis on $\Omega, x : G_3 \sqsubseteq G_4 \vdash t_3 : G'_3 \sqsubseteq t_4 : G'_4$, that $\Omega, x \mapsto c'' : G_3 \sqsubseteq G_4 \vdash c' : t'_3 \sqsubseteq t'_4$, where $c' = \text{initPT}(G'_3, G'_4)$ and $c'' = \text{initPT}(G_3, G_4)$. Thus, by Lemma B.113, we get that $G'_3 \sqsubseteq G'_4$, and therefore, $G_3- \longrightarrow G'_3 \sqsubseteq G_4- \longrightarrow G'_4$ because $G_3 \sqsubseteq G_4$.

Since $G_3- \longrightarrow G'_3 \sqsubseteq G_4- \longrightarrow G'_4$, $\varepsilon_3 = \text{reflEv}(G_3- \longrightarrow G'_3) = \text{initEv}(G_3- \longrightarrow G'_3, G_3- \longrightarrow G'_3)$, $\varepsilon_4 = \text{reflEv}(G_4- \longrightarrow G'_4) = \text{initEv}(G_4- \longrightarrow G'_4, G_4- \longrightarrow G'_4)$ and $c = c'' \longrightarrow c' = \text{initPT}(G_3- \longrightarrow G'_3, G_4- \longrightarrow G'_4) \longrightarrow \text{initPT}(G'_3, G'_4)$, by Lemma B.124, we know that $[c]\varepsilon_3 \sqsubseteq \varepsilon_4[c]$.

Thus, we get that

$$\Omega \vdash c : \text{reflEv}(G_3 \rightarrow G'_3) (\lambda x : G_3.t'_3) :: G_3 \rightarrow G'_3 \sqsubseteq \text{reflEv}(G_4 \rightarrow G'_4) (\lambda x : G_4.t'_4) :: G_4 \rightarrow G'_4$$

Thus, the result holds.

Case (Type abstractions). $t_1 = (\Lambda X.t_3)$ and $t_2 = (\Lambda X.t_4)$,

We know that

$$(\text{G}\Lambda) \frac{\Delta, X; \Gamma_1 \vdash t_3 : G_3 \quad \Delta \vdash \Gamma_1}{\Delta; \Gamma_1 \vdash \Lambda X.t_3 : \forall X.G_3} \quad (\text{G}\Lambda) \frac{\Delta, X; \Gamma_2 \vdash t_4 : G_4 \quad \Delta \vdash \Gamma_2}{\Delta; \Gamma_2 \vdash \Lambda X.t_4 : \forall X.G_4}$$

$$\frac{\Omega \vdash t_3 : G_3 \sqsubseteq t_4 : G_4}{\Omega \vdash \Lambda X.t_3 : \forall X.G_3 \sqsubseteq \Lambda X.t_4 : \forall X.G_4}$$

$$(\text{E}\Lambda) \frac{\Delta, X; \Gamma_1 \vdash t_3 : G_3 \rightsquigarrow t'_3 \quad \Delta \vdash \Gamma_1}{\Delta; \Gamma_1 \vdash \Lambda X.t_3 : \forall X.G_3 \rightsquigarrow \text{reflEv}(\forall X.G_3) (\Lambda X.t'_3) :: \forall X.G_3}$$

$$(\text{E}\Lambda) \frac{\Delta, X; \Gamma_2 \vdash t_4 : G_4 \rightsquigarrow t'_4 \quad \Delta \vdash \Gamma_2}{\Delta; \Gamma_2 \vdash \Lambda X.t_4 : \forall X.G_4 \rightsquigarrow \text{reflEv}(\forall X.G_4) (\Lambda X.t'_4) :: \forall X.G_4}$$

We are required to prove that

$$\Omega \vdash \mathbf{c} : \text{reflEv}(\forall X.G_3) (\Lambda X.t'_3) :: \forall X.G_3 \sqsubseteq \text{reflEv}(\forall X.G_4) (\Lambda X.t'_4) :: \forall X.G_4$$

where $\mathbf{c} = \text{initPT}(\forall X.G_3, \forall X.G_4) = \forall X.\text{initPT}(G_3, G_4)$.

We know, by the induction hypothesis on $\Omega \vdash t_3 : G_3 \sqsubseteq t_4 : G_4$, that $\Omega \vdash \mathbf{c}' : t'_3 \sqsubseteq t'_4$, where $\mathbf{c}' = \text{initPT}(G_3, G_4)$. Thus, by Lemma B.113, we get that $G_3 \sqsubseteq G_4$, and therefore, $\forall X.G_3 \sqsubseteq \forall X.G_4$.

Since $\forall X.G_3 \sqsubseteq \forall X.G_4$, $\varepsilon_3 = \text{reflEv}(\forall X.G_3) = \text{initEv}(\forall X.G_3, \forall X.G_3)$, $\varepsilon_4 = \text{reflEv}(\forall X.G_4) = \text{initEv}(\forall X.G_4, \forall X.G_4)$, $\mathbf{c} = \forall X.\mathbf{c}' = \forall X.\text{initPT}(G_3, G_4)$, by Lemma B.124, we know that $[\mathbf{c}]_{\varepsilon_3} \sqsubseteq \varepsilon_4[\mathbf{c}]$.

Thus, we get that $\Omega \vdash \mathbf{c} : \text{reflEv}(\forall X.G_3) (\Lambda X.t'_3) :: \forall X.G_3 \sqsubseteq \text{reflEv}(\forall X.G_4) (\Lambda X.t'_4) :: \forall X.G_4$, and the result holds.

Case (Function applications). $t_1 = t_3 t'_3$ and $t_2 = t_4 t'_4$.

We know that

$$\begin{array}{c} \Delta; \Gamma_1 \vdash t_3 : G_3 \quad \Delta; \Gamma_1 \vdash t'_3 : G'_3 \\ \text{(Gapp)} \frac{\text{dom}^\#(G_3) \sim G'_3}{\Delta; \Gamma_1 \vdash t_3 t'_3 : \text{cod}^\#(G_3)} \end{array} \quad \begin{array}{c} \Delta; \Gamma_2 \vdash t_4 : G_4 \quad \Delta; \Gamma_2 \vdash t'_4 : G'_4 \\ \text{(Gapp)} \frac{\text{dom}^\#(G_4) \sim G'_4}{\Delta; \Gamma_2 \vdash t_4 t'_4 : \text{cod}^\#(G_4)} \end{array}$$

Also, we know that

$$\frac{\Omega \vdash t_3 : G_3 \sqsubseteq t_4 : G_4 \quad \Omega \vdash t'_3 : G'_3 \sqsubseteq t'_4 : G'_4 \quad G_3 \rightarrow G_{31} \rightarrow G_{32} \quad G_4 \rightarrow G_{41} \rightarrow G_{42}}{\Omega \vdash t_3 t'_3 : G_{32} \sqsubseteq t_4 t'_4 : G_{42}}$$

$$\begin{array}{c} \Delta; \Gamma_1 \vdash t_3 : G_3 \rightsquigarrow t_{31} \quad \Delta; \Gamma_1 \vdash t'_3 : G'_3 \rightsquigarrow t'_{32} \quad G_3 \rightarrow G_{31} \rightarrow G_{32} \\ \text{(Eapp)} \frac{\varepsilon_3 = \text{initEv}(G_3, G_{31} \rightarrow G_{32}) \quad \varepsilon'_3 = \text{initEv}(G'_3, G_{31})}{\Delta; \Gamma_1 \vdash t_3 t'_3 : G_{32} \rightsquigarrow (\varepsilon_3 t_{31} :: G_{31} \rightarrow G_{32}) (\varepsilon'_3 t'_{32} :: G_{31})} \end{array}$$

$$\begin{array}{c} \Delta; \Gamma_2 \vdash t_4 : G_4 \rightsquigarrow t_{41} \quad \Delta; \Gamma_2 \vdash t'_4 : G'_4 \rightsquigarrow t'_{42} \quad G_4 \rightarrow G_{41} \rightarrow G_{42} \\ \text{(Eapp)} \frac{\varepsilon_4 = \text{initEv}(G_4, G_{41} \rightarrow G_{42}) \quad \varepsilon'_4 = \text{initEv}(G'_4, G_{41})}{\Delta; \Gamma_2 \vdash t_4 t'_4 : G_{42} \rightsquigarrow (\varepsilon_4 t_{41} :: G_{41} \rightarrow G_{42}) (\varepsilon'_4 t'_{42} :: G_{41})} \end{array}$$

We are required to prove that

$$\Omega \vdash \mathbf{c} : (\varepsilon_3 t_{31} :: G_{31} \rightarrow G_{32}) (\varepsilon'_3 t'_{32} :: G_{31}) \sqsubseteq (\varepsilon_4 t_{41} :: G_{41} \rightarrow G_{42}) (\varepsilon'_4 t'_{42} :: G_{41})$$

where $\mathbf{c} = \text{initPT}(G_{32}, G_{42})$.

We know, by the induction hypothesis on $\Omega \vdash t_3 : G_3 \sqsubseteq t_4 : G_4$, that $\Omega \vdash \mathbf{c}' : t_{31} \sqsubseteq t_{41}$, where $\mathbf{c}' = \text{initPT}(G_3, G_4)$. Thus, by Lemma B.113, we get that $G_3 \sqsubseteq G_4$. By Lemma B.131, we know that $G_{31} \rightarrow G_{32} \sqsubseteq G_{41} \rightarrow G_{42}$. By Lemma B.134, we get that $\text{dom}^\#(G_{31} \rightarrow G_{32}) = G_{31} \sqsubseteq G_{41} = \text{dom}^\#(G_{41} \rightarrow G_{42})$ and $\text{cod}^\#(G_{31} \rightarrow G_{32}) = G_{32} \sqsubseteq G_{42} = \text{cod}^\#(G_{41} \rightarrow G_{42})$.

Since $G_3 \sqsubseteq G_4$, $G_{31} \rightarrow G_{32} \sqsubseteq G_{41} \rightarrow G_{42}$, $\varepsilon_3 = \text{initEv}(G_3, G_{31} \rightarrow G_{32})$, $\varepsilon_4 = \text{initEv}(G_4, G_{41} \rightarrow G_{42})$, $\mathbf{c}' = \text{initPT}(G_3, G_4)$ and $\mathbf{c}_1 \rightarrow \mathbf{c}_2 = \text{initPT}(G_{31} \rightarrow G_{32}, G_{41} \rightarrow G_{42}) = \text{initPT}(G_{31}, G_{41}) \rightarrow \text{initPT}(G_{32}, G_{42})$, by Lemma B.124, we know that $[\mathbf{c}']_{\varepsilon_3} \sqsubseteq \varepsilon_4[\mathbf{c}_1 \rightarrow \mathbf{c}_2]$. Thus, we get that $\Omega \vdash \mathbf{c}_1 \rightarrow \mathbf{c}_2 : (\varepsilon_3 t_{31} :: G_{31} \rightarrow G_{32}) \sqsubseteq (\varepsilon_4 t_{41} :: G_{41} \rightarrow G_{42})$.

We know, by the induction hypothesis on $\Omega \vdash t'_3 : G'_3 \sqsubseteq t'_4 : G'_4$, that $\Omega \vdash c'' : t_{32} \sqsubseteq t_{42}$, where $c'' = \text{initPT}(G'_3, G'_4)$. Thus, by Lemma B.113, we get that $G'_3 \sqsubseteq G'_4$.

Since $G'_3 \sqsubseteq G'_4$, $G_{31} \sqsubseteq G_{41}$, $\varepsilon'_3 = \text{initEv}(G'_3, G_{31})$, $\varepsilon'_4 = \text{initEv}(G'_4, G_{41})$, $c'' = \text{initPT}(G'_3, G'_4)$ and $c_1 = \text{initPT}(G_{31}, G_{41})$, by Lemma B.124, we know that $[c'']\varepsilon'_3 \sqsubseteq \varepsilon'_4[c_1]$. Thus, we get that $\Omega \vdash c_1 : (\varepsilon'_3 t_{32} :: G_{31}) \sqsubseteq (\varepsilon'_4 t_{42} :: G_{41})$. Finally, the result holds.

Case (Type applications). $t_1 = t_3 [F]$ and $t_2 = t_4 [F]$.

We know that

$$(\text{GappG}) \frac{\Delta; \Gamma_1 \vdash t_3 : G_3 \quad \Delta \vdash F}{\Delta; \Gamma_1 \vdash t_3 [F] : \text{inst}^\sharp(G_3, F)} \quad (\text{GappG}) \frac{\Delta; \Gamma_2 \vdash t_4 : G_4 \quad \Delta \vdash F}{\Delta; \Gamma_2 \vdash t_4 [F] : \text{inst}^\sharp(G_4, F)}$$

Also, we know that

$$\frac{\Omega \vdash t_3 : G_3 \sqsubseteq t_4 : G_4 \quad G_3 \leq G_4}{G_3 \rightarrow \forall X. G'_3 \quad G_4 \rightarrow \forall X. G'_4} \quad \frac{\Delta; \Gamma_1 \vdash t_3 : G_3 \rightsquigarrow t'_3 \quad \Delta \vdash F \quad G_3 \rightarrow \forall X. G'_3}{\varepsilon_3 = \text{initEv}(G_3, \forall X. G'_3) \quad \varepsilon'_3 = \text{instEv}(G'_3, X, F)} \quad \frac{\Delta; \Gamma_2 \vdash t_3 [F] : G'_3[F/X] \setminus X \rightsquigarrow \varepsilon'_3((\varepsilon_3 t'_3 :: \forall X. G'_3) [F]) :: G'_3[F/X] \setminus X}{(\text{EappG})}$$

$$\frac{\Delta; \Gamma_2 \vdash t_4 : G_4 \rightsquigarrow t'_4 \quad \Delta \vdash F \quad G_4 \rightarrow \forall X. G'_4}{\varepsilon_4 = \text{initEv}(G_4, \forall X. G'_4) \quad \varepsilon'_4 = \text{instEv}(G'_4, X, F)} \quad \frac{\Delta; \Gamma_2 \vdash t_4 [F] : G'_4[F/X] \setminus X \rightsquigarrow \varepsilon'_4((\varepsilon_4 t'_4 :: \forall X. G'_4) [F]) :: G'_4[F/X] \setminus X}{(\text{EappG})}$$

We have to prove that

$$\Omega \vdash c : \varepsilon'_3((\varepsilon_3 t'_3 :: \forall X. G'_3) [F]) :: G'_3[F/X] \setminus X \sqsubseteq \varepsilon'_4((\varepsilon_4 t'_4 :: \forall X. G'_4) [F]) :: G'_4[F/X] \setminus X$$

where $c = \text{initPT}(G'_3[F/X] \setminus X, G'_4[F/X] \setminus X)$.

Since $G_3 \leq G_4$, by Lemma B.127, we get that $G_3 \sqsubseteq G_4$. Then, by Lemma B.133, we get that $\forall X. G'_3 \sqsubseteq \forall X. G'_4$ and $\forall X. G'_3 \leq \forall X. G'_4$.

We know, by the induction hypothesis on $\Omega \vdash t_3 : G_3 \sqsubseteq t_4 : G_4$ that $\Omega \vdash c' : t'_3 \sqsubseteq t'_4$, where $c' = \text{initPT}(G_3, G_4)$.

Since $G_3 \sqsubseteq G_4$, $\forall X. G'_3 \sqsubseteq \forall X. G'_4$, $\varepsilon_3 = \text{initEv}(G_3, \forall X. G'_3)$, $\varepsilon_4 = \text{initEv}(G_4, \forall X. G'_4)$, $c' = \text{initPT}(G_3, G_4)$ and $\forall X. c'' = \text{initPT}(\forall X. G'_3, \forall X. G'_4) = \forall X. \text{initPT}(G'_3, G'_4)$, by Lemma B.124, we know that $[c']\varepsilon_3 \sqsubseteq \varepsilon_4[\forall X. c'']$. Thus, we get that $\Omega \vdash \forall X. c'' : (\varepsilon_3 t'_3 :: \forall X. G'_3) \sqsubseteq (\varepsilon_4 t'_4 :: \forall X. G'_4)$, and, therefore, $\Omega \vdash c''' : (\varepsilon_3 t'_3 :: \forall X. G'_3) [F] \sqsubseteq (\varepsilon_4 t'_4 :: \forall X. G'_4) [F]$, where $c''' = c''[F/X]$.

Since we know that $\forall X. G'_3 \leq \forall X. G'_4$, $\varepsilon'_3 = \text{instEv}(G'_3, X, F)$, $\varepsilon'_4 = \text{instEv}(G'_4, X, F)$, $\forall X. c'' = \text{initPT}(\forall X. G'_3, \forall X. G'_4)$ and $c = \text{initPT}(G'_3[F/X] \setminus X, G'_4[F/X] \setminus X)$, by Lemma B.125, we get that $[c''[F/X]]\varepsilon_1 \sqsubseteq \varepsilon_2[c]$. Thus, the result follows immediately.

Case (Ascriptions). $t_1 = t_3 :: G_3$ and $t_2 = t_4 :: G_4$.

We know that

$$(\text{Gasc}) \frac{\Delta; \Gamma_1 \vdash t_3 : G'_3 \quad G'_3 \sim G_3}{\Delta; \Gamma_1 \vdash t_3 :: G_3} \quad (\text{Gasc}) \frac{\Delta; \Gamma_1 \vdash t_4 : G'_4 \quad G'_4 \sim G_4}{\Delta; \Gamma_2 \vdash t_4 :: G_4}$$

Also, we know that

$$\frac{\Omega \vdash t_3 : G'_3 \sqsubseteq t_4 : G'_4 \quad G_3 \sqsubseteq G_4}{\Omega \vdash t_3 :: G_3 : G_3 \sqsubseteq t_4 :: G_4 : G_4} \quad (\text{Easc}) \frac{\Delta; \Gamma_1 \vdash t_3 : G'_3 \rightsquigarrow t'_3 \quad \varepsilon_3 = \text{initEv}(G'_3, G_3)}{\Delta; \Gamma_1 \vdash t_3 :: G_3 : G_3 \rightsquigarrow \varepsilon_3 t'_3 :: G_3}$$

$$(\text{Easc}) \frac{\Delta; \Gamma_2 \vdash t_4 : G'_4 \rightsquigarrow t'_4 \quad \varepsilon_4 = \text{initEv}(G'_4, G_4)}{\Delta; \Gamma_2 \vdash t_4 :: G_4 : G_4 \rightsquigarrow \varepsilon_4 t'_4 :: G_4}$$

Therefore, we are required to prove that $\Omega \vdash c : (\varepsilon_3 t'_3 :: G_3) \sqsubseteq (\varepsilon_4 t'_4 :: G_4)$, where $c = \text{initPT}(G_3, G_4)$. Or what is the same, we are required to prove that $\Omega \vdash c' : t'_3 \sqsubseteq t'_4$, for some c' and $[c']\varepsilon_3 \sqsubseteq \varepsilon_4[c]$.

We know, by the induction hypothesis on $\Omega \vdash t_3 : G'_3 \sqsubseteq t_4 : G'_4$, that $\Omega \vdash c' : t'_3 \sqsubseteq t'_4$, where $c' = \text{initPT}(G'_3, G'_4)$. Thus, by Lemma B.113, we get that $G'_3 \sqsubseteq G'_4$. Since $G'_3 \sqsubseteq G'_4$, $G_3 \sqsubseteq G_4$, $\varepsilon_3 = \text{initEv}(G'_3, G_3)$, $\varepsilon_4 = \text{initEv}(G'_4, G_4)$, $c' = \text{initPT}(G'_3, G'_4)$ and $c = \text{initPT}(G_3, G_4)$, by Lemma B.124, we know that $[c']\varepsilon_3 \sqsubseteq \varepsilon_4[c]$. Thus, the result holds. \square

Theorem B.144 (Gradual guarantees) *Suppose* $\vdash t_1 : G_1 \sqsubseteq t_2 : G_2$ and $\vdash t_1 : G_1$.

1. $\vdash t_2 : G_2$ and $G_1 \sqsubseteq G_2$.
2. If $t_1 \Downarrow v_1$, then $t_2 \Downarrow v_2$ and $\vdash \text{initPT}(G_1, G_2) : v_1 \sqsubseteq v_2$.
If $t_1 \Uparrow$ then $t_2 \Uparrow$.

PROOF. We prove (1), directly, by Lemma B.111. Since $\vdash t_1 : G_1$, by Lemma B.111, we know that $\vdash t_2 : G_2$. Since $\vdash t_1 : G_1$ and $\vdash t_2 : G_2$, by Lemma 7.9, we know that $\vdash t_1 : G_1 \rightsquigarrow t'_1$ and $\vdash t_2 : G_2 \rightsquigarrow t'_2$. Thus, we prove (2), directly, by Lemma 7.10 and 6.14. \square

B.6.5 Parametricity

As the logical relations are not defined directly over $F^?$ terms, this definition establishes a “weaker” notion of parametricity than in $F_e^?$. In particular and as mentioned in Section ??, programs 1 and 2 of Table 6.1, $t_1 = (\Lambda X. \lambda x : ?_X. x :: X) [\text{Int}] \mathbf{42}$ and $t_2 = (\Lambda X. \lambda x : ?_X. x :: X) [\text{Bool}] \mathbf{42}$ behaves differently: the former reduces to $\mathbf{42}$ and the latter fails. Although the elaboration of $\Lambda X. \lambda x : ?_X. x :: X$, $t_V = \varepsilon'(\Lambda X. \lambda x : ?_X. \varepsilon_1 x :: X) :: \forall X. ?_X \rightarrow X$, is related to itself, the elaborations of t_1 and t_2 are not related. The respective elaborations are $(\varepsilon_2(t_V [\text{Int}] :: ? \rightarrow \text{Int}) (\varepsilon_{\mathbf{42}} :: ?))$ and $(\varepsilon_3(t_V [\text{Bool}] :: ? \rightarrow \text{Bool}) (\varepsilon_{\mathbf{42}} :: ?))$. As $\varepsilon = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\text{Int}})\}$, then $\varepsilon_{\mathbf{42}} :: ?_{X:\text{Int}}$ and $\varepsilon_{\mathbf{42}} :: ?_{X:\text{Bool}}$ are related at type $?_{X:X}$. Therefore $(t_V [\text{Int}]) (\varepsilon_{\mathbf{42}} :: ?_{X:\text{Int}})$ is related to $(t_V [\text{Bool}]) (\varepsilon_{\mathbf{42}} :: ?_{X:\text{Bool}})$ as both program will fail when trying to unseal the argument from X . The problem here are evidences ε_2 and ε_3 that automatically seals/unseals the argument: $(\varepsilon \ ; \ \text{dom}(\varepsilon_2))\mathbf{42} :: ?_{X:\text{Int}}$ is not related to $(\varepsilon \ ; \ \text{dom}(\varepsilon_3))\mathbf{42} :: ?_{X:\text{Bool}}$. This is because $\varepsilon \ ; \ \text{dom}(\varepsilon_2) = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\text{Int}}), (\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_X)\}$, but $\varepsilon \ ; \ \text{dom}(\varepsilon_3) = \{(\text{Int}, \text{refl}_{\text{Int}}, \text{inj}_{\text{Int}})\}$, and when combined with unsealing evidence $(\{(X, \text{inj}_X, X)\}[X \mapsto \text{Int}] = \{(\text{Int}, \text{inj}_X, \text{Int})\})$ and $(\{(X, \text{inj}_X, X)\}[X \mapsto \text{Bool}] = \{(\text{Bool}, \text{inj}_X, \text{refl}_{\text{Bool}})\})$ where $\vdash \{(X, \text{inj}_X, X)\} : X : X \rightarrow X$ the former succeeds but the latter fails. Note that this non-parametric behavior does not affect polymorphic functions whose type does not mention the unknown type, e.g. $t'_1 = (\Lambda X. \lambda x : X. x :: X) [\text{Int}] \mathbf{42}$ and $t'_2 = (\Lambda X. \lambda x : X. x :: X) [\text{Bool}] \mathbf{42}$ are not related as t'_2 does not typecheck.

B.6.6 Source-level parametric reasoning

Lemma B.145 *If $\neg?_X \in \mathbf{G}$, then $\text{schm}(\text{initEv}(\forall X.G, \forall X.G))[\mathbf{B}/\mathbf{X}] = \text{instEv}(\mathbf{G}, \mathbf{X}, \mathbf{B})$.*

Definition B.146

$$\begin{array}{ll}
?_{\delta'}[\mathbf{F}/\mathbf{X}]^? = ?_{\delta'}[\mathbf{F}/\mathbf{X}] & ?_{\delta'}[\cdot/\mathbf{X}] \setminus \mathbf{X}^? = ?_{\delta'}[\mathbf{X}/\mathbf{X}] \setminus \mathbf{X} \\
\mathbf{X}[\mathbf{F}/\mathbf{X}]^? = \mathbf{X} & \mathbf{X}[\cdot/\mathbf{X}] \setminus \mathbf{X}^? = \mathbf{X} \\
\mathbf{B}[\mathbf{F}/\mathbf{X}]^? = \mathbf{B} & \mathbf{B}[\cdot/\mathbf{X}] \setminus \mathbf{X}^? = \mathbf{B} \\
(\mathbf{G}_1 \rightarrow \mathbf{G}_2)[\mathbf{F}/\mathbf{X}]^? = \mathbf{G}_1[\mathbf{F}/\mathbf{X}]^? \rightarrow \mathbf{G}_2[\mathbf{F}/\mathbf{X}]^? & (\mathbf{G}_1 \rightarrow \mathbf{G}_2)[\cdot/\mathbf{X}] \setminus \mathbf{X}^? = \mathbf{G}_1[\cdot/\mathbf{X}] \setminus \mathbf{X}^? \rightarrow \mathbf{G}_2[\cdot/\mathbf{X}] \setminus \mathbf{X}^? \\
\forall \mathbf{Y}. \mathbf{G}[\mathbf{F}/\mathbf{X}]^? = \forall \mathbf{Y}. \mathbf{G}[\mathbf{F}/\mathbf{X}]^? & \forall \mathbf{Y}. \mathbf{G}[\cdot/\mathbf{X}] \setminus \mathbf{X}^? = \forall \mathbf{Y}. \mathbf{G}[\cdot/\mathbf{X}] \setminus \mathbf{X}^?
\end{array}$$

Definition B.147 $\text{instEv}^?(G, X, F) : G[\mathbf{F}/\mathbf{X}]^? \sim G[\cdot/\mathbf{X}] \setminus \mathbf{X}^?$

$$\begin{array}{ll}
\text{instEv}^?(B, X, F) = \text{instEv}(B, X, F) & \text{instEv}^?(G_1 \rightarrow G_2, X, F) = \text{instEv}^?(G_1, X, F) \rightarrow \text{instEv}^?(G_2, X, F) \\
\text{instEv}^?(X, X, F) = \text{initEv}(X, X) & \text{instEv}^?(?_{\delta}.G, X, F) = ?_{\delta}. \text{instEv}^?(G, X, F) \\
\text{instEv}^?(?_{\delta}, X, F) = \text{instEv}(?_{\delta}, X, F) &
\end{array}$$

Lemma B.148 $\text{instEv}(\mathbf{G}, \mathbf{X}, \mathbf{F}) = \text{instEv}^?(G, X, F)[\mathbf{F}/\mathbf{X}]$

Intuitively, a type abstraction f of type $\forall X.G$ behaves polymorphically if (1) X is not in the scope of unknown types in G , or (2) if X is in the scope of unknown types in G , is instantiated to the same type and using as relation the value interpretation of the instantiated type. Formally:

Lemma B.149

$$\frac{\Delta; \Gamma \vdash f : \forall X.G \quad \forall B_1, B_2, R \in \text{REL}[B_1, B_2] \quad \forall n, \rho, \Delta, ((n, \rho) \in \mathcal{D}[\Delta] \wedge (n, \gamma) \in \mathcal{G}_{\rho}[\Gamma])}{\begin{array}{l} X \notin [G]^? \\ \Delta; \Gamma \vdash f [B_i] : \text{inst}^{\#}(G, B_i) \rightsquigarrow t_i \quad G' = G[\cdot/\mathbf{X}] \setminus \mathbf{X}^? \\ (n, \rho_1(\gamma_1(t_1)), \rho_2(\gamma_2(t_2))) \in \mathcal{T}_{\rho, X \mapsto (B_1, B_2, R)}[G'] \end{array}}$$

PROOF.

1. Let $\varepsilon = \text{initEv}(\forall X.G, \forall X.G)$, and $\varepsilon'_i = \text{instEv}(G, X, B_i)$, $\Delta; \Gamma \vdash f : \forall X.G \rightsquigarrow \varepsilon f :: \forall X.G$.
2. Note that $\Delta; \Gamma \vdash f [B_i] : \text{inst}^{\#}(G, B_i) \rightsquigarrow \varepsilon'_i(\varepsilon f :: \forall X.G [B_i]) :: G[B_i/X] \setminus X$
3. Let $\rho' = \rho, X \mapsto (B_1, B_2, R)$.
4. Note that $\rho'_i(G') = \rho'_i(G[\cdot/X] \setminus X^?) = \rho_i(G[\cdot/X] \setminus X^?[B_i/X]) = \rho_i(G[B_i/X] \setminus X)$, therefore is well typed.
5. By Theorem 3.28, $(n, \rho_1(\gamma_1(\varepsilon f :: \forall X.G)), \rho_2(\gamma_2(\varepsilon f :: \forall X.G))) \in \mathcal{V}_{\rho}[\forall X.G]$
6. By (5), $(n, \rho_1(\gamma_1(\varepsilon f :: \forall X.G))[B_1], \rho_2(\gamma_2(\varepsilon f :: \forall X.G))[B_2]) \in \mathcal{T}_{\rho'}[G]$,
7. Let $\varepsilon_i = \rho_i(\varepsilon)$, $\varepsilon''_i = \rho_i(\varepsilon'_i)$, and $G_i = \rho_i(G)$.

8. If

$$\begin{aligned} \rho_1(\gamma_1(\varepsilon f :: \forall X. G))[\mathbf{B}_1] &\mapsto \text{schm}(\varepsilon_1)t'_1 :: \mathbf{G}_1[\mathbf{B}_1/\mathbf{X}] \\ &\xrightarrow{j} \text{schm}(\varepsilon_1)v_1 :: \mathbf{G}_1[\mathbf{B}_1/\mathbf{X}] \\ &\mapsto v'_1 \end{aligned}$$

then $\rho_2(\gamma_2(\varepsilon f :: \forall X. G))[\mathbf{B}_2] \xrightarrow{*} \text{schm}(\varepsilon_2)[\mathbf{B}_2/\mathbf{X}]v_2 :: \mathbf{G}_2[\mathbf{B}_2/\mathbf{X}] \mapsto v'_2$, and $\triangleright^{j+2} (n, v'_1, v'_2) \in \mathcal{V}_{\rho'}[\mathbf{G}]$

9. Then $\rho_1(\gamma_1(\varepsilon'_i(\varepsilon f :: \forall X. G [\mathbf{B}_i]) :: \mathbf{G}[\mathbf{B}_i/\mathbf{X}] \setminus \mathbf{X})) \mapsto \varepsilon''_i(\text{schm}(\varepsilon_i)[\mathbf{B}_i/\mathbf{X}]t'_i :: \mathbf{G}_i[\mathbf{B}_i/\mathbf{X}] :: \mathbf{G}_i[\mathbf{B}_i/\mathbf{X}] \setminus \mathbf{X})$

10. If $\neg?_{\mathbf{X}} \in \mathbf{G}$

(a) By Lemma B.145 $\varepsilon'_i = \text{schm}(\varepsilon_i)[\mathbf{B}_i/\mathbf{X}]$, therefore $\varepsilon''_i = \text{schm}(\varepsilon_i)[\mathbf{B}_i/\mathbf{X}]$. Also by Lemma MT
▶ TODO ◀ $\mathbf{G}[\mathbf{B}_i/\mathbf{X}] = \mathbf{G}[\mathbf{B}_i/\mathbf{X}] \setminus \mathbf{X}$, then

$$\begin{aligned} &\varepsilon''_1(\varepsilon'_1 t'_1 :: \mathbf{G}_1[\mathbf{B}_1/\mathbf{X}] :: \mathbf{G}_1[\mathbf{B}_1/\mathbf{X}]) \\ &\xrightarrow{j} \varepsilon''_1(\varepsilon'_1 v_1 :: \mathbf{G}_1[\mathbf{B}_1/\mathbf{X}] :: \mathbf{G}_1[\mathbf{B}_1/\mathbf{X}]) \\ &\xrightarrow{2} v'_1 \end{aligned} \quad (\text{By Lemma 6.7})$$

Similarly $\varepsilon''_2(\varepsilon'_2 t'_2 :: \mathbf{G}_2[\mathbf{B}_2/\mathbf{X}] :: \mathbf{G}_2[\mathbf{B}_2/\mathbf{X}]) \xrightarrow{*} v'_2$, and by weakening $\triangleright^{j+3} (n, v'_1, v'_2) \in \mathcal{V}_{\rho'}[\mathbf{G}]$. As $\neg?_{\mathbf{X}} \in \mathbf{G}$ then $\mathbf{G} = \mathbf{G}'$ and the result holds. □

Lemma B.150 *If $\vdash f : \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X}$, $\vdash v : \mathbf{B}$ and $t = f [\mathbf{B}] v$, then $t \Downarrow v$ with $\vdash v : \mathbf{B} \rightsquigarrow v$, or $t \Downarrow \text{error}$, or $t \Uparrow$.*

PROOF.

- We know that $\vdash f [\mathbf{B}] v : \mathbf{B} \rightsquigarrow (\varepsilon_2((\varepsilon_1 f :: \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X}) [\mathbf{B}]) :: ? \rightarrow \mathbf{B}) (\varepsilon_3 v :: ?)$, where
 - $\vdash f : \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X} \rightsquigarrow f$.
 - $\vdash v : \mathbf{B} \rightsquigarrow v$.
 - $\varepsilon_1 : \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X} \sim \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X}$ and $\varepsilon_1 = \text{initEv}(\forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X}, \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X})$.
 - $\varepsilon_2 : ?_{\mathbf{X}:\mathbf{B}} \rightarrow \mathbf{B} \sim ? \rightarrow \mathbf{B}$ and $\varepsilon_2 = \text{instEv}(?_{\mathbf{X}} \rightarrow \mathbf{X}, \mathbf{X}, \mathbf{B}) = \{(\mathbf{B} \rightarrow \mathbf{B}, \text{inj}_{\mathbf{X}} \rightarrow \mathbf{B}, \text{inj}_{\mathbf{B}} \rightarrow \mathbf{B}), (? \rightarrow \mathbf{B}, \text{inj}_{? \rightarrow \mathbf{B}}, \text{inj}_{? \rightarrow \mathbf{B}})\}$.
 - $\varepsilon_3 : \mathbf{B} \sim ? = \text{initEv}(\mathbf{B}, ?) = \{(\mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{B}})\}$.
- By Lemma 7.9 and the Fundamental Property 3.28, we know that $\vdash \varepsilon_1 f :: \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X} \preceq \varepsilon_1 f :: \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X} : \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X}$ and $\vdash \varepsilon_3 v :: ? \preceq \varepsilon_3 v :: ? : ?$.
- We instantiate $\vdash \varepsilon_1 f :: \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X} \preceq \varepsilon_1 f :: \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X} : \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X}$ with $n, \rho = \emptyset$ and $\gamma = \emptyset$. Therefore, we know that $(n, \varepsilon_1 f :: \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X}, \varepsilon_1 f :: \forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X}) \in \mathcal{T}_{\rho}[\forall X. ?_{\mathbf{X}} \rightarrow \mathbf{X}]$.

- If $(\varepsilon_2((\varepsilon_1 \mathbf{f} :: \forall \mathbf{X} . ?_{\mathbf{X}} \rightarrow \mathbf{X}) [\mathbf{B}]) :: ? \rightarrow \mathbf{B}) (\varepsilon_3 \mathbf{v} :: ?) \xrightarrow{i} \mathbf{v}^*$, then we know by the operational semantics that

$$\begin{aligned}
& (\varepsilon_2((\varepsilon_1 \mathbf{f} :: \forall \mathbf{X} . ?_{\mathbf{X}} \rightarrow \mathbf{X}) [\mathbf{B}]) :: ? \rightarrow \mathbf{B}) (\varepsilon_3 \mathbf{v} :: ?) \\
& \xrightarrow{i_1} (\varepsilon_2(\mathbf{v}_{\mathbf{f}} [\mathbf{B}]) :: ? \rightarrow \mathbf{B}) (\varepsilon_3 \mathbf{v} :: ?) \\
& \xrightarrow{i_2} (\varepsilon_2 \mathbf{v}'_{\mathbf{f}} :: ? \rightarrow \mathbf{B}) (\varepsilon_3 \mathbf{v} :: ?) \quad \mathbf{v}'_{\mathbf{f}} = \varepsilon_{\mathbf{v}'_{\mathbf{f}}}(\lambda \mathbf{x} : \mathbf{G} . \mathbf{t}) :: ?_{\mathbf{X} : \mathbf{B}} - \\
& \xrightarrow{0} ((\varepsilon_{\mathbf{v}'_{\mathbf{f}}} \circ \varepsilon_2)(\lambda \mathbf{x} : \mathbf{G} . \mathbf{t}) :: ? \rightarrow \mathbf{B}) (\varepsilon_3 \mathbf{v} :: ?) \\
& \xrightarrow{0} ((\varepsilon_{\mathbf{v}'_{\mathbf{f}}} \circ \varepsilon_2)(\lambda \mathbf{x} : \mathbf{G} . \mathbf{t}) :: ? \rightarrow \mathbf{B}) \mathbf{v}' \quad \mathbf{v}' = \varepsilon_{\mathbf{v}'} \mathbf{u} \\
& \xrightarrow{1} \text{cod}(\varepsilon_{\mathbf{v}'_{\mathbf{f}}} \circ \varepsilon_2)(\mathbf{t}[\text{cod}(\varepsilon_{\mathbf{v}'_{\mathbf{f}}} \circ \varepsilon_2) \mathbf{u} :: \mathbf{G}] / \mathbf{x}] :: \mathbf{B} \\
& \quad \text{cod}(\varepsilon_2)(\text{cod}(\varepsilon_{\mathbf{v}'_{\mathbf{f}}})(\mathbf{t}[\text{cod}(\varepsilon_{\mathbf{v}'_{\mathbf{f}}} \circ \varepsilon_2) \mathbf{u} :: \mathbf{G}] / \mathbf{x}] :: \mathbf{G}') :: \mathbf{B} \\
& \xrightarrow{i_3} \text{cod}(\varepsilon_2)(\text{cod}(\varepsilon_{\mathbf{v}'_{\mathbf{f}}}) \mathbf{v}_{\mathbf{t}} :: \mathbf{G}') :: \mathbf{B} \\
& \xrightarrow{0} \text{cod}(\varepsilon_2) \mathbf{v}'_{\mathbf{t}} :: \mathbf{B} \\
& \xrightarrow{0} \mathbf{v}^*
\end{aligned}$$

where $i = i_1 + i_2 + i_3 + 1$.

- If $\varepsilon_1 \mathbf{f} :: \forall \mathbf{X} . ?_{\mathbf{X}} \rightarrow \mathbf{X} \xrightarrow{*} \mathbf{error}$ or $\varepsilon_1 \mathbf{f} :: \forall \mathbf{X} . ?_{\mathbf{X}} \rightarrow \mathbf{X}$ diverges, then the result holds immediately. Otherwise, if $\varepsilon_1 \mathbf{f} :: \forall \mathbf{X} . ?_{\mathbf{X}} \rightarrow \mathbf{X} \xrightarrow{i_1} \mathbf{v}_{\mathbf{f}}$, then we know that $\triangleright^{i_1} (n, \mathbf{v}_{\mathbf{f}}, \mathbf{v}_{\mathbf{f}}) \in \mathcal{V}_{\rho}[\forall \mathbf{X} . ?_{\mathbf{X}} \rightarrow \mathbf{X}]$.
- We instantiate $\triangleright^{i_1} (n, \mathbf{v}_{\mathbf{f}}, \mathbf{v}_{\mathbf{f}}) \in \mathcal{V}_{\rho}[\forall \mathbf{X} . ?_{\mathbf{X}} \rightarrow \mathbf{X}]$ with $\mathbf{B}_1 = \mathbf{B}_2 = \mathbf{B}$ and $R = \{(n', \mathbf{v}, \mathbf{v}) \mid n' \in \mathbb{N}\} \in \text{REL}[\mathbf{B}, \mathbf{B}]$. Therefore, we get that $\triangleright^{i_1} (n, \mathbf{v}_{\mathbf{f}} [\mathbf{B}], \mathbf{v}_{\mathbf{f}} [\mathbf{B}]) \in \mathcal{T}_{\rho'}[?_{\mathbf{X}} \rightarrow \mathbf{X}]$, where $\rho' = \rho; \mathbf{X} \mapsto (\mathbf{B}, \mathbf{B}, R)$.
- If $\mathbf{v}_{\mathbf{f}} [\mathbf{B}] \xrightarrow{*} \mathbf{error}$ or diverges, then the result holds immediately. Otherwise, $\mathbf{v}_{\mathbf{f}} [\mathbf{B}] \xrightarrow{i_2} \mathbf{v}'_{\mathbf{f}}$ and we instantiate $\triangleright^{i_1} (n, \mathbf{v}_{\mathbf{f}} [\mathbf{B}], \mathbf{v}_{\mathbf{f}} [\mathbf{B}]) \in \mathcal{T}_{\rho'}[?_{\mathbf{X}} \rightarrow \mathbf{X}]$ with i_2 and $\mathbf{v}'_{\mathbf{f}}$, obtaining that $\triangleright^{i_1+i_2} (n, \mathbf{v}'_{\mathbf{f}}, \mathbf{v}'_{\mathbf{f}}) \in \mathcal{V}_{\rho'}[?_{\mathbf{X}} \rightarrow \mathbf{X}]$.
- We instantiate $\vdash \varepsilon_3 \mathbf{v} :: ? \preceq \varepsilon_3 \mathbf{v} :: ? : ?$ with $n - (i_1 + i_2 + 1)$. If $\varepsilon_3 \mathbf{v} :: ? \xrightarrow{*} \mathbf{error}$ or diverges the result holds immediately. Otherwise, $\varepsilon_3 \mathbf{v} :: ? \xrightarrow{0} \mathbf{v}'$ and $\triangleright^{i_1+i_2+1} (n, \mathbf{v}', \mathbf{v}') \in \mathcal{V}_{\rho}[?]$.
- Observe that $\varepsilon_{\mathbf{v}'} = \{(\mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{B}})\}$ and $\text{dom}(\varepsilon_2) = \{(\mathbf{B}, \text{inj}_{\mathbf{B}}, \text{inj}_{\mathbf{X}}), (? , \text{inj}_{?}, \text{inj}_{?})\}$. Therefore, $\varepsilon_{\mathbf{v}'} \circ \text{dom}(\varepsilon_2) = \{(\mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{X}}), (\mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{B}})\}$. Let us define $\delta = \{\mathbf{X} : \mathbf{X}\}$. By case analysis on $\vdash \varepsilon : \delta \rightarrow \mathbf{G}_{\mathbf{R}}$ we get that
 - $\vdash \{(\mathbf{B}, \text{inj}_{\mathbf{B}}, \text{refl}_{\mathbf{B}})\} : \delta \rightarrow \mathbf{B}$ and $\rho'_i(\varepsilon) = \varepsilon = \{(\mathbf{B}, \text{inj}_{\mathbf{B}}, \text{refl}_{\mathbf{B}})\}$. Therefore, we get that $\rho'_i(\varepsilon) (\varepsilon_{\mathbf{v}'} \circ \text{dom}(\varepsilon_2) \mathbf{u} :: ?_{\mathbf{X} : \mathbf{B}}) :: \mathbf{B} \xrightarrow{0} \mathbf{v}$ and $\triangleright^{i_1+i_2+1} (n, \mathbf{v}, \mathbf{v}) \in \mathcal{V}_{\rho'}[\mathbf{B}]$. Therefore, $\triangleright^{i_1+i_2+1} (n, \rho'_1(\varepsilon) (\varepsilon_{\mathbf{v}'} \circ \text{dom}(\varepsilon_2) \mathbf{u} :: ?_{\mathbf{X} : \mathbf{B}}) :: \mathbf{B}, \rho'_2(\varepsilon) (\varepsilon_{\mathbf{v}'} \circ \text{dom}(\varepsilon_2) \mathbf{u} :: ?_{\mathbf{X} : \mathbf{B}}) :: \mathbf{B}) \in \mathcal{T}_{\rho'}[\mathbf{B}]$.
 - $\vdash \{(\mathbf{X}, \text{inj}_{\mathbf{X}}, \text{refl}_{\mathbf{X}})\} : \delta \rightarrow \mathbf{X}$, $\mathbf{X} : \mathbf{X} \in \delta$ and $\delta \vdash \mathbf{X}$ and $\rho'_i(\varepsilon) = \{(\mathbf{B}, \text{inj}_{\mathbf{X}}, \text{refl}_{\mathbf{B}})\}$. Therefore, we get that $\rho'_i(\varepsilon) (\varepsilon_{\mathbf{v}'} \circ \text{dom}(\varepsilon_2) \mathbf{u} :: ?_{\mathbf{X} : \mathbf{B}}) :: \mathbf{B} \xrightarrow{0} \mathbf{v}$ and $\triangleright^{i_1+i_2+1} (n, \mathbf{v}, \mathbf{v}) \in \mathcal{V}_{\rho'}[\mathbf{X}]$. Observe that $\mathcal{V}_{\rho'}[\mathbf{X}] = \rho' . R(\mathbf{X}) = R$. Therefore, $\triangleright^{i_1+i_2+1} (n, \rho'_1(\varepsilon) (\varepsilon_{\mathbf{v}'} \circ \text{dom}(\varepsilon_2) \mathbf{u} :: ?_{\mathbf{X} : \mathbf{B}}) :: \mathbf{B}, \rho'_2(\varepsilon) (\varepsilon_{\mathbf{v}'} \circ \text{dom}(\varepsilon_2) \mathbf{u} :: ?_{\mathbf{X} : \mathbf{B}}) :: \mathbf{B}) \in \mathcal{T}_{\rho'}[\mathbf{X}]$.
 - $\vdash \{(?_{\delta} \rightarrow ?_{\delta}, \text{inj}_{\rightarrow}, \text{inj}_{?} \rightarrow \text{inj}_{?})\} : \delta \rightarrow ?_{\delta} \rightarrow ?_{\delta}$. We get that $\triangleright^{i_1+i_2+1} (n, \rho'_1(\varepsilon) (\varepsilon_{\mathbf{v}'} \circ \text{dom}(\varepsilon_2) \mathbf{u} :: ?_{\mathbf{X} : \mathbf{B}}) :: \rho'_1(?_{\delta} \rightarrow ?_{\delta}), \rho'_2(\varepsilon) (\varepsilon_{\mathbf{v}'} \circ \text{dom}(\varepsilon_2) \mathbf{u} :: ?_{\mathbf{X} : \mathbf{B}}) :: \rho'_2(?_{\delta} \rightarrow ?_{\delta})) \in \mathcal{T}_{\rho'}[?_{\delta} \rightarrow ?_{\delta}]$, since $\rho'_i(\varepsilon) (\varepsilon_{\mathbf{v}'} \circ \text{dom}(\varepsilon_2) \mathbf{u} :: ?_{\mathbf{X} : \mathbf{B}}) :: \rho'_i(?_{\delta} \rightarrow ?_{\delta}) \xrightarrow{0} \mathbf{error}$.

- $\vdash \{(\delta \times \delta, \text{inj}_\times, \text{inj}_\gamma \times \text{inj}_\gamma)\} : \delta \rightarrow \delta \times \delta$. We get that $\triangleright^{i_1+i_2+1} (n, \rho'_1(\varepsilon) (\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}) :: \rho'_1(\delta \times \delta), \rho'_2(\varepsilon) (\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}) :: \rho'_2(\delta \times \delta)) \in \mathcal{T}_{\rho'}[\delta \times \delta]$, since $\rho'_i(\varepsilon) (\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}) :: \rho'_i(\delta \times \delta) \xrightarrow{0} \mathbf{error}$.
- $\vdash \{(\forall \mathbf{Y}.\delta, \mathbf{Y}, \text{inj}_{\forall}, \forall \mathbf{Y}.\text{inj}_\gamma)\} : \delta \rightarrow \forall \mathbf{Y}.\delta, \mathbf{Y}$. We get that $\triangleright^{i_1+i_2+1} (n, \rho'_1(\varepsilon) (\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}) :: \rho'_1(\forall \mathbf{Y}.\delta, \mathbf{Y}), \rho'_2(\varepsilon) (\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}) :: \rho'_2(\forall \mathbf{Y}.\delta, \mathbf{Y})) \in \mathcal{T}_{\rho'}[\forall \mathbf{Y}.\delta, \mathbf{Y}]$, since $\rho'_i(\varepsilon) (\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}) :: \rho'_i(\forall \mathbf{Y}.\delta, \mathbf{Y}) \xrightarrow{0} \mathbf{error}$.

Thus, we get that $\triangleright^{i_1+i_2+1} (n, \varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}, \varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}) \in \mathcal{V}_{\rho'}[\mathbf{X}:\mathbf{X}]$.

- We instantiate $\triangleright^{i_1+i_2} (n, \mathbf{v}'_f, \mathbf{v}'_f) \in \mathcal{V}_{\rho'}[\mathbf{X} \rightarrow \mathbf{X}]$ with $i_1 + i_2$, $\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}$ and $\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}$ such that $\triangleright^{i_1+i_2+1} (n, \varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}, \varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}) \in \mathcal{V}_{\rho'}[\mathbf{X}:\mathbf{X}]$. Thus, we get that $\triangleright^{i_1+i_2} (n, \mathbf{v}'_f (\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}), \mathbf{v}'_f (\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B})) \in \mathcal{T}_{\rho'}[\mathbf{X}]$.
- If $\mathbf{v}'_f (\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}) \xrightarrow{*} \mathbf{error}$ or diverges, then the result holds immediately. Otherwise, $\mathbf{v}'_f (\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}) \xrightarrow{i_3} \mathbf{v}'_t$ and we instantiate $\triangleright^{i_1+i_2} (n, \mathbf{v}'_f (\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B}), \mathbf{v}'_f (\varepsilon_{\mathbf{V}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?\mathbf{X}:\mathbf{B})) \in \mathcal{T}_{\rho'}[\mathbf{X}]$ with i_3 and \mathbf{v}'_t , obtaining that $\triangleright^{i_1+i_2+i_3} (n, \mathbf{v}'_t, \mathbf{v}'_t) \in \mathcal{V}_{\rho'}[\mathbf{X}]$. Since $\mathcal{V}_{\rho'}[\mathbf{X}] = \rho'.R(\mathbf{X}) = R$ we get that $\mathbf{v}'_t = \mathbf{v}$.
- Observe that $\text{cod}(\varepsilon_2) = \{(\mathbf{B}, \mathbf{B}, \mathbf{B})\}$. Thus, $\text{cod}(\varepsilon_2) \mathbf{v}'_t :: \mathbf{B} \xrightarrow{0} \mathbf{v}$ and $\mathbf{v}^* = \mathbf{v}$. Finally, the result holds.

□

Lemma B.151 *If $\vdash f : \forall \mathbf{X}.\delta \rightarrow \mathbf{X}$, $\vdash v : \mathbf{B}$ and $t = f[\mathbf{B}] v$, then $t \Downarrow \mathbf{error}$, or $t \Uparrow$.*

PROOF. By *Reductio ad absurdum*. Let us suppose that $t \Downarrow \mathbf{v}^*$. We are going to prove that we get a contradiction.

- We know that $\vdash f[\mathbf{B}] v : \mathbf{B} \rightsquigarrow (\varepsilon_2((\varepsilon_1 f :: \forall \mathbf{X}.\delta \rightarrow \mathbf{X})[\mathbf{B}]) :: ? \rightarrow \mathbf{B}) (\varepsilon_3 v :: ?)$, where
 - $\vdash f : \forall \mathbf{X}.\delta \rightarrow \mathbf{X} \rightsquigarrow f$.
 - $\vdash v : \mathbf{B} \rightsquigarrow v$.
 - $\varepsilon_1 : \forall \mathbf{X}.\delta \rightarrow \mathbf{X} \sim \forall \mathbf{X}.\delta \rightarrow \mathbf{X}$ and $\varepsilon_1 = \text{initEv}(\forall \mathbf{X}.\delta \rightarrow \mathbf{X}, \forall \mathbf{X}.\delta \rightarrow \mathbf{X})$.
 - $\varepsilon_2 : ? \rightarrow \mathbf{B} \sim ? \rightarrow \mathbf{B}$ and $\varepsilon_2 = \text{instEv}(? \rightarrow \mathbf{X}, \mathbf{X}, \mathbf{B}) = \{(? \rightarrow \mathbf{B}, \text{inj}_\gamma \rightarrow \mathbf{B}, \text{inj}_\gamma \rightarrow \mathbf{B})\}$.
 - $\varepsilon_3 : \mathbf{B} \sim ? = \text{initEv}(\mathbf{B}, ?) = \{(\mathbf{B}, \mathbf{B}, \text{inj}_{\mathbf{B}})\}$.
- By Lemma 7.9 and the Fundamental Property 3.28, we know that $\vdash \varepsilon_1 f :: \forall \mathbf{X}.\delta \rightarrow \mathbf{X} \preceq \varepsilon_1 f :: \forall \mathbf{X}.\delta \rightarrow \mathbf{X} : \forall \mathbf{X}.\delta \rightarrow \mathbf{X}$ and $\vdash \varepsilon_3 v :: ? \preceq \varepsilon_3 v :: ? : ?$.
- We instantiate $\vdash \varepsilon_1 f :: \forall \mathbf{X}.\delta \rightarrow \mathbf{X} \preceq \varepsilon_1 f :: \forall \mathbf{X}.\delta \rightarrow \mathbf{X} : \forall \mathbf{X}.\delta \rightarrow \mathbf{X}$ with $n, \rho = \emptyset$ and $\gamma = \emptyset$. Therefore, we know that $(n, \varepsilon_1 f :: \forall \mathbf{X}.\delta \rightarrow \mathbf{X}, \varepsilon_1 f :: \forall \mathbf{X}.\delta \rightarrow \mathbf{X}) \in \mathcal{T}_\rho[\forall \mathbf{X}.\delta \rightarrow \mathbf{X}]$.
- If $(\varepsilon_2((\varepsilon_1 f :: \forall \mathbf{X}.\delta \rightarrow \mathbf{X})[\mathbf{B}]) :: ? \rightarrow \mathbf{B}) (\varepsilon_3 v :: ?) \xrightarrow{i} \mathbf{v}^*$, then we know by the operational semantics that

$$\begin{aligned}
& (\varepsilon_2 ((\varepsilon_1 \mathbf{f} :: \forall \mathbf{X} . ? \rightarrow \mathbf{X}) [\mathbf{B}]) :: ? \rightarrow \mathbf{B}) (\varepsilon_3 \mathbf{v} :: ?) \\
& \xrightarrow{i_1} (\varepsilon_2 (\mathbf{v}_f [\mathbf{B}]) :: ? \rightarrow \mathbf{B}) (\varepsilon_3 \mathbf{v} :: ?) \\
& \xrightarrow{i_2} (\varepsilon_2 \mathbf{v}'_f :: ? \rightarrow \mathbf{B}) (\varepsilon_3 \mathbf{v} :: ?) \qquad \mathbf{v}'_f = \varepsilon_{\mathbf{v}'_f} (\lambda \mathbf{x} : \mathbf{G} . \mathbf{t}) :: ? \rightarrow \mathbf{B} \\
& \xrightarrow{0} ((\varepsilon_{\mathbf{v}'_f} \ ; \ \varepsilon_2) (\lambda \mathbf{x} : \mathbf{G} . \mathbf{t}) :: ? \rightarrow \mathbf{B}) (\varepsilon_3 \mathbf{v} :: ?) \\
& \xrightarrow{0} ((\varepsilon_{\mathbf{v}'_f} \ ; \ \varepsilon_2) (\lambda \mathbf{x} : \mathbf{G} . \mathbf{t}) :: ? \rightarrow \mathbf{B}) \mathbf{v}' \qquad \mathbf{v}' = \varepsilon_{\mathbf{v}'} \mathbf{u} :: ? \\
& \xrightarrow{1} \text{cod}(\varepsilon_{\mathbf{v}'_f} \ ; \ \varepsilon_2) (\mathbf{t} [((\varepsilon_{\mathbf{v}'_f} \ ; \ \text{dom}(\varepsilon_{\mathbf{v}'_f}) \ ; \ \varepsilon_2)) \mathbf{u} :: \mathbf{G}] / \mathbf{x}) :: \mathbf{B} \\
& \qquad \text{cod}(\varepsilon_2) (\text{cod}(\varepsilon_{\mathbf{v}'_f}) (\mathbf{t} [(((\varepsilon_{\mathbf{v}'_f} \ ; \ \text{dom}(\varepsilon_2)) \ ; \ \text{dom}(\varepsilon_{\mathbf{v}'_f})) \mathbf{u} :: \mathbf{G}) / \mathbf{x}] :: \mathbf{G}') :: \mathbf{B} \\
& \xrightarrow{i_3} \text{cod}(\varepsilon_2) (\text{cod}(\varepsilon_{\mathbf{v}'_f}) \mathbf{v}_t :: \mathbf{G}') :: \mathbf{B} \\
& \xrightarrow{0} \text{cod}(\varepsilon_2) \mathbf{v}'_t :: \mathbf{B} \\
& \xrightarrow{0} \mathbf{v}^*
\end{aligned}$$

where $i = i_1 + i_2 + i_3 + 1$.

- Since $\varepsilon_1 \mathbf{f} :: \forall \mathbf{X} . ? \rightarrow \mathbf{X} \xrightarrow{i_1} \mathbf{v}_f$, then we know that $\triangleright^{i_1} (n, \mathbf{v}_f, \mathbf{v}_f) \in \mathcal{V}_\rho [\forall \mathbf{X} . ? \rightarrow \mathbf{X}]$.
- We instantiate $\triangleright^{i_1} (n, \mathbf{v}_f, \mathbf{v}_f) \in \mathcal{V}_\rho [\forall \mathbf{X} . ? \rightarrow \mathbf{X}]$ with $\mathbf{B}_1 = \mathbf{B}_2 = \mathbf{B}$ and $R = \emptyset \in \text{REL}[\mathbf{B}, \mathbf{B}]$. Therefore, we get that $\triangleright^{i_1} (n, \mathbf{v}_f [\mathbf{B}], \mathbf{v}_f [\mathbf{B}]) \in \mathcal{T}_{\rho'} [? \rightarrow \mathbf{X}]$, where $\rho' = \rho; \mathbf{X} \mapsto (\mathbf{B}, \mathbf{B}, R)$.
- Since $\mathbf{v}_f [\mathbf{B}] \xrightarrow{i_2} \mathbf{v}'_f$, we instantiate $\triangleright^{i_1} (n, \mathbf{v}_f [\mathbf{B}], \mathbf{v}_f [\mathbf{B}]) \in \mathcal{T}_{\rho'} [? \rightarrow \mathbf{X}]$ with i_2 and \mathbf{v}'_f , obtaining that $\triangleright^{i_1+i_2} (n, \mathbf{v}'_f, \mathbf{v}'_f) \in \mathcal{V}_{\rho'} [? \rightarrow \mathbf{X}]$.
- We instantiate $\vdash \varepsilon_3 \mathbf{v} :: ? \preceq \varepsilon_3 \mathbf{v} :: ? : ?$ with $n - (i_1 + i_2 + 1)$. Since $\varepsilon_3 \mathbf{v} :: ? \xrightarrow{0} \mathbf{v}'$, we get that $\triangleright^{i_1+i_2+1} (n, \mathbf{v}', \mathbf{v}') \in \mathcal{V}_\rho [?]$. By the Ascription Lemma 6.10, with evidence $\text{dom}(\varepsilon_2) : ? \sim ?$, we get that $\triangleright^{i_1+i_2+1} (n, \varepsilon_{\mathbf{v}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?, \varepsilon_{\mathbf{v}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?) \in \mathcal{V}_{\rho'} [?]$.
- We instantiate $\triangleright^{i_1+i_2} (n, \mathbf{v}'_f, \mathbf{v}'_f) \in \mathcal{V}_{\rho'} [? \rightarrow \mathbf{X}]$ with $i_1 + i_2$, $\varepsilon_{\mathbf{v}'_f} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?$ and $\varepsilon_{\mathbf{v}'} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?$ such that $\triangleright^{i_1+i_2+1} (n, \varepsilon_{\mathbf{v}'_f} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?, \varepsilon_{\mathbf{v}'_f} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?) \in \mathcal{V}_{\rho'} [?]$. Thus, we get that $\triangleright^{i_1+i_2} (n, \mathbf{v}'_f (\varepsilon_{\mathbf{v}'_f} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?), \mathbf{v}'_f (\varepsilon_{\mathbf{v}'_f} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?)) \in \mathcal{T}_{\rho'} [\mathbf{X}]$.
- Since $\mathbf{v}'_f (\varepsilon_{\mathbf{v}'_f} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?) \xrightarrow{i_3} \mathbf{v}'_t$, we instantiate $\triangleright^{i_1+i_2} (n, \mathbf{v}'_f (\varepsilon_{\mathbf{v}'_f} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?), \mathbf{v}'_f (\varepsilon_{\mathbf{v}'_f} \ ; \ \text{dom}(\varepsilon_2) \mathbf{u} :: ?)) \in \mathcal{T}_{\rho'} [\mathbf{X}]$ with i_3 and \mathbf{v}'_t , obtaining that $\triangleright^{i_1+i_2+i_3} (n, \mathbf{v}'_t, \mathbf{v}'_t) \in \mathcal{V}_{\rho'} [\mathbf{X}]$. Since $\mathcal{V}_{\rho'} [\mathbf{X}] = \rho' . R(\mathbf{X}) = R = \emptyset$ we get a contradiction and the result holds immediately.

□

Universal type is degenerate

In this section, we attempt to prove that the counterexample of Devriese et al. (2018) does not apply to $F_\varepsilon^?$.

Fully-abstract embedding? Jacobs et al. (2021) propose the fully-abstract embedding of a statically-typed language into its gradually-typed counterpart as a criteria to assess gradual languages. Informally, this means that terms contextually equivalent in the static language must also

be contextually equivalent in the gradual language. Such a result has been disproved for λB via the following counterexample (Devriese et al., 2018). Consider type $\mathbf{Univ} \triangleq \exists Y. \forall X. (X \rightarrow Y) \times (Y \rightarrow X)$ and the next two terms:

$$\begin{aligned} \mathbf{t}_s &\triangleq \lambda x : \mathbf{Univ}. \text{unpack } x \text{ as } \langle Y, x' \rangle \text{ in let } x'' : (\mathbf{Unit} \rightarrow Y) \times (Y \rightarrow \mathbf{Unit}) = x' [\mathbf{Unit}] \text{ in } \pi_2(x'') (\pi_1(x'') \text{ unit}) \\ \mathbf{t}_u &\triangleq \lambda x : \mathbf{Univ}. \text{unpack } x \text{ as } \langle Y, x' \rangle \text{ in let } x'' : (\mathbf{Unit} \rightarrow Y) \times (Y \rightarrow \mathbf{Unit}) = x' [\mathbf{Unit}] \text{ in } \pi_2(x'') (\pi_1(x'') \text{ unit}); \omega_{\mathbf{Unit}} \end{aligned}$$

where $\omega_{\mathbf{Unit}}$ is the always-diverging term of type \mathbf{Unit} . Intuitively, \mathbf{t}_s and \mathbf{t}_u are almost the same function, but the $\omega_{\mathbf{Unit}}$ in \mathbf{t}_u makes it always diverge when invoked. Devriese et al. prove that the invocation $\pi_2(x'') (\pi_1(x'') \text{ unit})$ must diverge for all System F inhabitants of \mathbf{Univ} , so that the terms \mathbf{t}_s and \mathbf{t}_u are contextually equivalent.

In λB (and GSF) these two terms are no longer contextually equivalent: there is a context in which the application of \mathbf{t}_s terminates. Essentially, the globally scoped sealing used in these languages allows constructing an inhabitant of \mathbf{Univ} which instantiates Y to the unknown type and does not make the term \mathbf{t}_s diverge.

While proving fully-abstract embedding for $F_\varepsilon^?$ is future work, we can already show that the lexically scoped sealing of $F_\varepsilon^?$ breaks this counterexample. The following lemma states that the elaborations of \mathbf{t}_s and \mathbf{t}_u are related.

Lemma B.152 *If $\vdash \mathbf{t}_s : \mathbf{Univ} \rightarrow \mathbf{Unit} \rightsquigarrow \mathbf{t}_s$ and $\vdash \mathbf{t}_u : \mathbf{Univ} \rightarrow \mathbf{Unit} \rightsquigarrow \mathbf{t}_u$, then $\vdash \mathbf{t}_s \preceq \mathbf{t}_u : \mathbf{Univ} \rightarrow \mathbf{Unit}$.*

We currently do not have an existential type, but we can use the version of the example that uses the standard encoding of the existential type (copied with permission from the auxiliary material of Devriese et al. (2018)):

$$\begin{aligned} \mathbf{Univ}' &= \forall Z. (\forall Y. (\forall \mathbf{X}. (X \rightarrow Y) \times (Y \rightarrow X)) \rightarrow Z) \rightarrow Z \\ \mathbf{t}'_u &= \lambda x : \mathbf{Univ}'. x \text{ Unit } (\lambda Y. \lambda x' : (\forall \mathbf{X}. (X \rightarrow Y) \times (Y \rightarrow X)). \\ &\quad \text{let } x'' : (\mathbf{Unit} \rightarrow \mathbf{Y}) \times (Y \rightarrow \mathbf{Unit}) = x' \text{ Unit in } \pi_2(x'') (\pi_1(x'') \text{ unit})) \\ \mathbf{t}'_d &= \lambda x : \mathbf{Univ}'. x \text{ Unit } (\lambda Y. \lambda x' : (\forall \mathbf{X}. (X \rightarrow Y) \times (Y \rightarrow X)). \\ &\quad \text{let } x'' : (\mathbf{Unit} \rightarrow \mathbf{Y}) \times (Y \rightarrow \mathbf{Unit}) = x' \text{ Unit in } (\pi_2(x'') (\pi_1(x'') \text{ unit}); \omega_{\mathbf{Unit}})) \end{aligned}$$

The following is a proof sketch that shows that \mathbf{t}'_u is contextually equivalent to \mathbf{t}'_d . The idea is to prove that the two terms are logically related and use a soundness lemma about the LR to obtain contextual equivalence. In this report, we haven't actually proven such a soundness lemma but it should be easy to prove based on the compatibility lemmas that we have used for proving the fundamental theorem (Theorem 3.28).

Lemma B.153 *$(n, \mathbf{t}'_u, \mathbf{t}'_d) \in \mathcal{T}_\emptyset[\mathbf{Univ}' \rightarrow \mathbf{Unit}]$ for arbitrary n .*

PROOF SKETCH. Take $n_1 \leq n$ and $\mathbf{v}_1, \mathbf{v}'_1 \in \mathcal{V}_\emptyset[\mathbf{Univ}']$, then it suffices to prove that $(n', \mathbf{t}'_u \mathbf{v}_1, \mathbf{t}'_d \mathbf{v}_2) \in \mathcal{T}_\rho[\mathbf{Unit}]$.

Clearly, if $\mathbf{t}'_u \mathbf{v}_1 \mapsto^* \mathbf{v}_2$, then we must have that $\mathbf{v}_1 \text{ Unit } \mathbf{v}_2 \mapsto^* \mathbf{v}_3$ for

$$\mathbf{v}_2 = (\lambda Y. \lambda x' : (\forall \mathbf{X}. (X \rightarrow Y) \times (Y \rightarrow X)). \text{let } x'' : (\mathbf{Unit} \rightarrow \mathbf{Y}) \times (Y \rightarrow \mathbf{Unit}) = x' \text{ Unit in } \pi_2(x'') (\pi_1(x'') \text{ unit}))$$

and similarly if $t'_d \mathbf{v}'_1 \mapsto * \mathbf{v}'_3$.

It suffices to prove that for $R_\emptyset = \emptyset \in \text{REL}[\text{Unit}, \text{Unit}]$, $\mathcal{V}_\rho \llbracket (n_1, \mathbf{v}_2, \mathbf{v}'_2) \rrbracket \in \mathcal{V}_{Z \mapsto R_\emptyset} \llbracket \forall Y. (\forall \mathbf{X}. (X \rightarrow Y) \times (Y \rightarrow X)) \rightarrow Z \rrbracket$.

To prove this, take $\vdash \tau_Y$ and $\vdash \tau'_Y$ and $R_Y \in \text{REL}[\tau_Y, \tau'_Y]$, then it suffices to prove that for $n_2 \leq n_1$

$$(n_2, \mathbf{v}_2 \tau_Y, \mathbf{v}'_2 \tau'_Y) \in \mathcal{J}_{Z \mapsto R_\emptyset, Y \mapsto R_Y} \llbracket (\forall \mathbf{X}. (X \rightarrow Y) \times (Y \rightarrow X)) \rightarrow Z \rrbracket$$

To prove this, it suffices to prove that $(n_2, v_4, v'_4) \in \mathcal{V}_{Z \mapsto R_\emptyset, Y \mapsto R_Y} \llbracket (\forall \mathbf{X}. (X \rightarrow Y) \times (Y \rightarrow X)) \rightarrow Z \rrbracket$ for

$$\mathbf{v}_4 = \lambda \mathbf{x}' : (\forall \mathbf{X}. (\mathbf{X} \rightarrow \tau_Y) \times (\tau_Y \rightarrow \mathbf{X})). \text{let } \mathbf{x}'' : (\text{Unit} \rightarrow \tau_Y) \times (\tau_Y \rightarrow \text{Unit}) = \mathbf{x}' \text{ Unit in } \pi_2(\mathbf{x}'') (\pi_1(\mathbf{x}'') \text{ unit})$$

and v'_4 similarly.

To prove this, take some $n_3 \leq n_2$ and $(n_3, v_5, v'_5) \in \mathcal{V}_{Z \mapsto R_\emptyset, Y \mapsto R_Y} \llbracket \forall \mathbf{X}. (X \rightarrow Y) \times (Y \rightarrow X) \rrbracket$, then it suffices to prove that $(n_3, v_4, v_5, v'_4, v'_5) \in \mathcal{J}_{Z \mapsto R_\emptyset, Y \mapsto R_Y} \llbracket Z \rrbracket$.

It suffices to prove that $(n_3, t_6, t'_6) \in \mathcal{J}_{Z \mapsto R_\emptyset, Y \mapsto R_Y} \llbracket Z \rrbracket$ for

$$t_6 = \text{let } x'' : (\text{Unit} \rightarrow \tau_Y) \times (\tau_Y \rightarrow \text{Unit}) = v_5 \text{ Unit in } \pi_2(x'') (\pi_1(x'') \text{ unit})$$

Now, take $R_\top = (n, \text{tt}, \text{tt}) \in \text{REL}[\text{Unit}, \text{Unit}]$ and we have that $(n_3, v_5 \text{ Unit}, v'_5 \text{ Unit}) \in \mathcal{J}_{Z \mapsto R_\emptyset, Y \mapsto R_Y, X \mapsto R_X} \llbracket (X \rightarrow Y) \times (Y \rightarrow X) \rrbracket$ for both $R_X = R_\top$ and R_\emptyset . Therefore, when $v_5 \text{ Unit} \mapsto * v_7$ then also $v'_5 \text{ Unit} \mapsto * v'_7$, and we have that $(n_4, v_7, v'_7) \in \mathcal{V}_{Y \mapsto R_Y, X \mapsto R_X} \llbracket (X \rightarrow Y) \times (Y \rightarrow X) \rrbracket$ for both choices for R_X and a smaller n_4 . It follows that $(n_4, \pi_1(v_7), \pi_1(v'_7)) \in \mathcal{J}_{Y \mapsto R_Y, X \mapsto R_\top} \llbracket \mathbf{X} \rightarrow \mathbf{Y} \rrbracket$ and $(n_4, \pi_2(v_7), \pi_2(v'_7)) \in \mathcal{J}_{Y \mapsto R_Y, X \mapsto R_\emptyset} \llbracket \mathbf{X} \rightarrow \mathbf{Y} \rrbracket$.

Since $(n_4, \text{unit}, \text{unit}) \in \mathcal{V}_{Y \mapsto R_Y, X \mapsto R_\top} \llbracket \mathbf{X} \rrbracket = R_\top$, we have that $(n_4, \pi_1(v_7) \text{ unit}, \pi_1(v'_7) \text{ unit}) \in \mathcal{J}_{Y \mapsto R_Y, X \mapsto R_\top} \llbracket \mathbf{Y} \rrbracket$. Since $\mathcal{J}_{Y \mapsto R_Y, X \mapsto R_\top} \llbracket \mathbf{Y} \rrbracket = \mathcal{J}_{Y \mapsto R_Y, X \mapsto R_\emptyset} \llbracket \mathbf{Y} \rrbracket$, it further follows that $(n_4, \pi_2(v_7) (\pi_1(v_7) \text{ unit}), \pi_2(v'_7) (\pi_1(v'_7) \text{ unit})) \in \mathcal{J}_{Y \mapsto R_Y, X \mapsto R_\emptyset} \llbracket \mathbf{X} \rrbracket$. Finally, because $\mathcal{V}_{Y \mapsto R_Y, X \mapsto R_\emptyset} \llbracket \mathbf{X} \rrbracket = \emptyset$, this means that $\pi_2(v_7) (\pi_1(v_7) \text{ unit})$ and $\pi_2(v'_7) (\pi_1(v'_7) \text{ unit})$ must diverge. The remaining proof goal then follows easily from some reasoning about the operational behavior of t_6 and t'_6 . \square

B.7 Examples

Table B.1 summarizes a set of examples extracted from the most recent papers addressing gradual parametricity (λB (Ahmed et al., 2017), System F_G (Igarashi et al., 2017a), GSF (Toro et al., 2019) and PolyG $^\nu$ (New et al., 2020)). The fourth column refers to the program identifier in our Table 6.1, indicating that the programs are similar or are displayed to show the same challenge in gradual parametricity. It is important to note that the programs in Table B.1 are programs of a gradual version of System F (except for p5 y p6). Therefore, in order to obtain the expected behavior in λB , PolyG $^\nu$ and $F_\varepsilon^?$, we have to insert corresponding casts, seal/unseal terms and evidences, respectively. Programs p5 and p6 use explicit sealing, so they are only relevant for the languages PolyG $^\nu$ and $F_\varepsilon^?$ (in the case of PolyG $^\nu$, $seal_X$ represent sealing terms, and in the case of $F_\varepsilon^?$ sealing evidences).

Table B.2 summarizes the different approaches to gradual parametricity, giving the behavior of each program from Table B.1.

Id	Language	Program	IdOT	Description
p1	PolyG ^ν	$(\lambda X.\lambda x:X.x :: ? :: \text{Bool}) \text{ [Bool] true}$	5	VPR
p2	PolyG ^ν	$((\lambda X.\lambda x:X.x) :: \forall X.X \rightarrow ? \text{ [Int] } 1) + 2$	4	SNE
p3	PolyG ^ν	$(\lambda X.\lambda x:X.\text{true}) :: \forall X.? \rightarrow \text{Bool} \text{ [Int] } 3$	1	NPB
p4	PolyG ^ν	$(\lambda X.\lambda x:X.\text{true}) :: \forall X.? \rightarrow \text{Bool} \text{ [Bool] } 3$	3	NPB/RTI
p5	PolyG ^ν	$(\lambda X.\lambda x:X.\text{true}) :: \forall X.? \rightarrow \text{Bool} \text{ [Int] } \text{seal}_X(3)$	1	NPB
p6	PolyG ^ν	$(\lambda X.\lambda x:X.\text{true}) :: \forall X.? \rightarrow \text{Bool} \text{ [Bool] } \text{seal}_X(3)$	3	NPB/RTI
p7	GSF	$(\lambda g:\forall X.X \rightarrow X.g \text{ [Int] } 10) ((\lambda X.\lambda x:X.x) :: \forall X.? \rightarrow ?)$		ELP
p8	GSF	$(\lambda g:\forall X.X \rightarrow X.g \text{ [Int] } 10) ((\lambda X.\lambda x:?x) :: \forall X.? \rightarrow ?)$		ELP
p9	GSF	$(\lambda g:\forall X.X \rightarrow X.g \text{ [Int] } 10) ((\lambda X.\lambda x:?x+1) :: \forall X.? \rightarrow ?)$	5	VPR
p10	GSF	$((\lambda X.\lambda x:X.x) :: \forall X.X \rightarrow ? \text{ [Int] } 1) + 2$		SNE
p11	GSF	$(\lambda X.\lambda x:X.x) :: \forall X.? \rightarrow ? \text{ [Int] } \text{true}$	3	RTI
p12	System F _d	$(\lambda X.\lambda x:\text{Int}.x :: ? :: X) \text{ [Int] } 1$	5	VPR
p13	System F _d	$(\lambda X.\lambda x:\text{Int}.x :: ? :: X) \text{ [Bool] } 1$		VPR
p14	System F _d	$(\lambda x:\forall X.X \rightarrow X.x) (\lambda X.\lambda x:X.x) \text{ [Int] } 42$		ELP
p15	System F _d	$(\lambda x:\forall X.? \rightarrow X.x) (\lambda X.\lambda x:X.x) \text{ [Int] } 42$		ELP
p16	λB	$(\lambda X.\lambda Y.\lambda p:X \times Y.\langle \pi_2(p), \pi_1(p) \rangle) \text{ [Int] [Bool] } \langle 1, \text{true} \rangle$		VPR
p17	λB	$(\lambda X.\lambda Y.\lambda p:X \times Y.\langle \pi_2(p), \pi_1(p) \rangle) \text{ [Bool] [Int] } \langle \text{true}, 1 \rangle$		VPR
p18	λB	$((\lambda X.\lambda Y.\lambda p:?p) :: \forall X.\forall Y.(X \times Y) \rightarrow (Y \times X)) \text{ [Int] [Int] } \langle 1, 2 \rangle$		VPR

Table B.1: Examples.

Id: Program identifier. IdOT: Program identifier in our table. VPR: Disguised violations of parametricity are detected at runtime. SNE: Seals do not escape the lexical scope of type abstraction. NPB: Shallow non-parametric behavior. RTI: Type instantiations on a polymorphic use of `?` are respected. ELP: External loss of precision.

Id	λB	System F _G	GSF	PolyG ^ν	F [?]	F [?] _ε
p1	error	error	error	error	error	error
p2	error	error	3	3	3	3
p3*	true	true	true	error	true	error
p4*	true	true	error	error	error	error
p5*	-	-	-	true	-	true
p6*	-	-	-	ill-typed	-	ill-typed
p7	error	error	10	10	10	10
p8	error	error	10	10	10	10
p9	error	error	error	error	error	error
p10	error	error	3	3	3	3
p11	true	true	error	error	error	error
p12	error	error	error	error	error	error
p13	error	error	error	error	error	error
p14	42	42	42	42	42	42
p15	error	error	42	42	42	42
p16	⟨true, 1⟩	⟨true, 1⟩	⟨true, 1⟩	⟨true, 1⟩	⟨true, 1⟩	⟨true, 1⟩
p17	⟨1, true⟩	⟨1, true⟩	⟨1, true⟩	⟨1, true⟩	⟨1, true⟩	⟨1, true⟩
p18	error	error	error	error	error	error

Table B.2: Comparison of approaches to gradual parametricity.

(*): Program p3, p4, p5 and p6 only insert seal/unseal terms (PolyG^ν) or seal/unseal evidences (F[?]_ε) if it is explicit using the seal syntax.

$\delta ::= \delta, X : X \mid \cdot$
 $F ::= B \mid X$ (simple types)
 $G ::= F \mid G \rightarrow G \mid G \times G \mid \forall X. G \mid ?_\delta$ (source gradual types)
 $t ::= b \mid \lambda x : G. t \mid \Lambda X. t \mid x \mid \langle t, t \rangle \mid t t \mid \pi_i(t) \mid t [F] \mid t :: G$ (source gradual terms)

$\Delta; \Gamma \vdash t : G$ Term typing

$$\begin{array}{c}
 G_x \frac{x : G \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x : G} \quad G_b \frac{\theta(b) = B \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash b : B} \quad G_\lambda \frac{\Delta; \Gamma, x : G_1 \vdash t : G_2}{\Delta; \Gamma \vdash \lambda x : G_1. t : G_1 \rightarrow G_2} \\
 G_\times \frac{\Delta; \Gamma \vdash t_1 : G_1 \quad \Delta; \Gamma \vdash t_2 : G_2}{\Delta; \Gamma \vdash \langle t_1, t_2 \rangle : G_1 \times G_2} \quad G_\Lambda \frac{\Delta, X; \Gamma \vdash t : G \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash \Lambda X. t : \forall X. G} \\
 G_{\text{asc}} \frac{\Delta; \Gamma \vdash t : G' \quad \Delta \vdash G \quad G' \sim G}{\Delta; \Gamma \vdash t :: G : G} \\
 G_{\text{app}} \frac{\Delta; \Gamma \vdash t_1 : G_1 \quad \Delta; \Gamma \vdash t_2 : G_2 \quad \text{dom}^\sharp(G_1) \sim G_2}{\Delta; \Gamma \vdash t_1 t_2 : \text{cod}^\sharp(G_1)} \quad G_{\text{pair}_i} \frac{\Delta; \Gamma \vdash t : G}{\Delta; \Gamma \vdash \pi_i(t) : \text{proj}_i^\sharp(G)} \\
 G_{\text{app}G} \frac{\Delta; \Gamma \vdash t : G \quad \Delta \vdash F}{\Delta; \Gamma \vdash t [F] : \text{inst}^\sharp(G, F)}
 \end{array}$$

$\text{dom}^\sharp : \text{OTYPE} \rightarrow \text{OTYPE}$ $\text{cod}^\sharp : \text{OTYPE} \rightarrow \text{OTYPE}$ $\text{proj}_i^\sharp : \text{OTYPE} \rightarrow \text{OTYPE}$ $\text{inst}^\sharp : \text{OTYPE}^2 \rightarrow \text{OTYPE}$
 $\text{dom}^\sharp(G_1 \rightarrow G_2) = G_1$ $\text{cod}^\sharp(G_1 \rightarrow G_2) = G_2$ $\text{proj}_i^\sharp(G_1 \times G_2) = G_i$ $\text{inst}^\sharp(\forall X. G, G') = G[G'/X]$
 $\text{dom}^\sharp(?_\delta) = ?_\delta$ $\text{cod}^\sharp(?_\delta) = ?_\delta$ $\text{proj}_i^\sharp(?_\delta) = ?_\delta$ $\text{inst}^\sharp(?_\delta, G') = ?_\delta$
 $\text{dom}^\sharp(G)$ undefined o/w $\text{cod}^\sharp(G)$ undefined o/w $\text{proj}_i^\sharp(G)$ undefined o/w $\text{inst}^\sharp(G, G')$ undefined o/w

$G \sim G$ Type Consistency

$$\begin{array}{c}
 \frac{}{B \sim B} \quad \frac{}{X \sim X} \quad \frac{G_1 \sim G'_1 \quad G_2 \sim G'_2}{G_1 \times G_2 \sim G'_1 \times G'_2} \quad \frac{G'_1 \sim G_1 \quad G_2 \sim G'_2}{G_1 \rightarrow G_2 \sim G'_1 \rightarrow G'_2} \\
 \frac{G \sim G'}{\forall X. G \sim \forall X. G'} \quad \frac{}{B \sim ?_\delta} \quad \frac{}{?_\delta \sim B} \quad \frac{X \in \delta}{X \sim ?_\delta} \quad \frac{X \in \delta}{?_\delta \sim X} \\
 \frac{G_1 \sim ?_\delta \quad G_2 \sim ?_\delta}{G_1 \rightarrow G_2 \sim ?_\delta} \quad \frac{G_1 \sim ?_\delta \quad G_2 \sim ?_\delta}{G_1 \times G_2 \sim ?_\delta} \quad \frac{G \sim ?_\delta, X}{\forall X. G \sim ?_\delta} \quad \frac{?_\delta \sim G_1 \quad ?_\delta \sim G_2}{?_\delta \sim G_1 \rightarrow G_2} \\
 \frac{?_\delta \sim G_1 \quad ?_\delta \sim G_2}{?_\delta \sim G_1 \times G_2} \quad \frac{?_\delta, X \sim G}{?_\delta \sim \forall X. G} \quad \frac{}{?_\delta \sim ?_{\delta'}}
 \end{array}$$

$G \sqsubseteq G$ Type Precision

$$\begin{array}{c}
 \sqsubseteq_B \frac{}{B \sqsubseteq B} \quad \sqsubseteq_X \frac{}{X \sqsubseteq X} \quad \sqsubseteq_{\rightarrow} \frac{G_1 \sqsubseteq G_2 \quad G'_1 \sqsubseteq G'_2}{G_1 \rightarrow G'_1 \sqsubseteq G_2 \rightarrow G'_2} \quad \sqsubseteq_{\times} \frac{G_1 \sqsubseteq G_2 \quad G'_1 \sqsubseteq G'_2}{G_1 \times G'_1 \sqsubseteq G_2 \times G'_2} \\
 \sqsubseteq_{\forall} \frac{G_1 \sqsubseteq G_2}{\forall X. G_1 \sqsubseteq \forall X. G_2} \quad \sqsubseteq_{B?} \frac{}{B \sqsubseteq ?_\delta} \quad \sqsubseteq_{X?} \frac{X \in \delta}{X \sqsubseteq ?_\delta} \quad \sqsubseteq_{\rightarrow?} \frac{G_1 \rightarrow G_2 \sqsubseteq ?_\delta \rightarrow ?_\delta}{G_1 \rightarrow G_2 \sqsubseteq ?_\delta} \\
 \sqsubseteq_{\times?} \frac{G_1 \times G_2 \sqsubseteq ?_\delta \times ?_\delta}{G_1 \times G_2 \sqsubseteq ?_\delta} \quad \sqsubseteq_{\forall?} \frac{\forall X. G \sqsubseteq \forall X. ?_\delta, X}{\forall X. G \sqsubseteq ?_\delta} \quad \sqsubseteq_{?} \frac{\delta \sqsubseteq \delta'}{?_\delta \sqsubseteq ?_{\delta'}}
 \end{array}$$

$\Delta; \Gamma \vdash t : G \rightsquigarrow t$ **Elaboration rules**

$$\begin{array}{c}
\text{Eb} \frac{\theta(b) = B \quad \Delta \vdash \Gamma \quad \varepsilon = \text{reflEv}(B)}{\Delta; \Gamma \vdash b : B \rightsquigarrow \varepsilon b :: B} \qquad \text{Ex} \frac{x : G \in \Gamma \quad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x : G \rightsquigarrow x} \\
\\
\text{E}_\times \frac{\Delta; \Gamma \vdash t_1 : G_1 \rightsquigarrow t'_1 \quad \Delta; \Gamma \vdash t_2 : G_2 \rightsquigarrow t'_2}{\Delta; \Gamma \vdash \langle t_1, t_2 \rangle : G_1 \times G_2 \rightsquigarrow \langle t'_1, t'_2 \rangle} \\
\\
\text{E}_\lambda \frac{\Delta; \Gamma, x : G_1 \vdash t : G_2 \rightsquigarrow t' \quad \varepsilon = \text{reflEv}(G_1 \rightarrow G_2)}{\Delta; \Gamma \vdash \lambda x : G_1. t : G_1 \rightarrow G_2 \rightsquigarrow \varepsilon (\lambda x : G_1. t') :: G_1 \rightarrow G_2} \\
\\
\text{E}_\Lambda \frac{\Delta, X; \Gamma \vdash t : G \rightsquigarrow t' \quad \Delta \vdash \Gamma \quad \varepsilon = \text{reflEv}(\forall X. G)}{\Delta; \Gamma \vdash \Lambda X. t : \forall X. G \rightsquigarrow \varepsilon (\Lambda X. t') :: \forall X. G} \\
\\
\text{Easc} \frac{\Delta; \Gamma \vdash t : G' \rightsquigarrow t' \quad \Delta \vdash G \quad \varepsilon = \text{initEv}(G', G)}{\Delta; \Gamma \vdash t : G : G \rightsquigarrow \varepsilon t' :: G} \\
\\
\text{Eapp} \frac{\Delta; \Gamma \vdash t_1 : G_1 \rightsquigarrow t'_1 \quad \Delta; \Gamma \vdash t_2 : G_2 \rightsquigarrow t'_2 \quad G_1 \rightarrow G_{11} \rightarrow G_{12} \quad \varepsilon_1 = \text{initEv}(G_1, G_{11} \rightarrow G_{12}) \quad \varepsilon_2 = \text{initEv}(G_2, G_{11})}{\Delta; \Gamma \vdash t_1 t_2 : G_{12} \rightsquigarrow (\varepsilon_1 t'_1 :: G_{11} \rightarrow G_{12}) (\varepsilon_2 t'_2 :: G_{11})} \\
\\
\text{EappG} \frac{\Delta; \Gamma \vdash t : G \rightsquigarrow t' \quad \Delta \vdash F \quad G \rightarrow \forall X. G' \quad \varepsilon_1 = \text{initEv}(G, \forall X. G') \quad \varepsilon_2 = \text{instEv}(G', X, F)}{\Delta; \Gamma \vdash t [F] : G'[F/X] \setminus X \rightsquigarrow \varepsilon_2 ((\varepsilon_1 t' :: \forall X. G') [F']) :: G'[F'/X] \setminus X} \\
\\
\text{Epairi} \frac{\Delta; \Gamma \vdash t : G \rightsquigarrow t' \quad G \rightarrow G_1 \times G_2 \quad \varepsilon = \text{initEv}(G, G_1 \times G_2)}{\Delta; \Gamma \vdash \pi_i(t) : G_i \rightsquigarrow \pi_i(\varepsilon t' :: G_1 \times G_2)}
\end{array}$$

$G \rightarrow G$ **Type matching**

$$\begin{array}{l}
G_1 \rightarrow G_2 \rightarrow G_1 \rightarrow G_2 \quad \forall X. G \rightarrow \forall X. G \quad G_1 \times G_2 \rightarrow G_1 \times G_2 \quad ?_\delta \rightarrow ?_\delta \rightarrow ?_\delta \quad ?_\delta \rightarrow \forall X. ?_{\delta, X} \\
?_\delta \rightarrow ?_\delta \times ?_\delta
\end{array}$$

Figure B.7: Translation from $F^?$ to $F_e^?$

$G \leq G$ Strict Type Precision

$$\begin{array}{c}
\leq_B \frac{}{B \leq B} \quad \leq_X \frac{}{X \leq X} \quad \leq_{\rightarrow} \frac{G_1 \leq G_2 \quad G'_1 \leq G'_2}{G_1 \rightarrow G'_1 \leq G_2 \rightarrow G'_2} \quad \leq_{\times} \frac{G_1 \leq G_2 \quad G'_1 \leq G'_2}{G_1 \times G'_1 \leq G_2 \times G'_2} \\
\leq_{\forall} \frac{G_1 \leq G_2}{\forall X. G_1 \leq \forall X. G_2} \quad \leq_{B?} \frac{}{B \leq ?_{\delta}} \quad \leq_{X?} \frac{X \in \delta}{X \leq ?_{\delta}} \quad \leq_{?} \frac{\delta \subseteq \delta'}{?_{\delta} \leq ?_{\delta'}}
\end{array}$$

$\Omega \vdash t : G \sqsubseteq t : G$ Term precision

$$\begin{array}{c}
\sqsubseteq_b \frac{}{\Omega \vdash b : B \sqsubseteq b : B} \quad \sqsubseteq_x \frac{x : G_1 \sqsubseteq G_2 \in \Omega}{\Omega \vdash x : G_1 \sqsubseteq x : G_2} \\
\sqsubseteq_{\lambda} \frac{\Omega, x : G_1 \sqsubseteq G_2 \vdash t_1 : G'_1 \sqsubseteq t_2 : G'_2 \quad G_1 \sqsubseteq G_2}{\Omega \vdash \lambda x : G_1. t_1 : G_1 \rightarrow G'_1 \sqsubseteq \lambda x : G_2. t_2 : G_2 \rightarrow G'_2} \\
\sqsubseteq_{\times} \frac{\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2 \quad \Omega \vdash t'_1 : G'_1 \sqsubseteq t'_2 : G'_2}{\Omega \vdash \langle t_1, t'_1 \rangle : G_1 \times G'_1 \sqsubseteq \langle t_2, t'_2 \rangle : G_2 \times G'_2} \quad \sqsubseteq_{\wedge} \frac{\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2}{\Omega \vdash \Lambda X. t_1 : \forall X. G_1 \sqsubseteq \Lambda X. t_2 : \forall X. G_2} \\
\sqsubseteq_{\text{asc}} \frac{\Omega \vdash t_1 : G'_1 \sqsubseteq t_2 : G_2 \quad G_1 \sqsubseteq G_2}{\Omega \vdash t_1 :: G_1 : G_1 \sqsubseteq t_2 :: G_2 : G_2} \quad \sqsubseteq_{\text{app}} \frac{\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2 \quad \Omega \vdash t'_1 : G'_1 \sqsubseteq t'_2 : G'_2}{G_1 \rightarrow G_{11} \rightarrow G_{12} \quad G_2 \rightarrow G_{21} \rightarrow G_{22}}{\Omega \vdash t_1 t'_1 : G_{12} \sqsubseteq t_2 t'_2 : G_{22}} \\
\sqsubseteq_{\text{pairi}} \frac{\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2}{G_1 \rightarrow G_{11} \times G_{12} \quad G_2 \rightarrow G_{21} \times G_{22}}{\Omega \vdash \pi_i(t_1) : G_{1i} \sqsubseteq \pi_i(t_2) : G_{2i}} \\
\sqsubseteq_{\text{appG}} \frac{\Omega \vdash t_1 : G_1 \sqsubseteq t_2 : G_2 \quad G_1 \leq G_2}{G_1 \rightarrow \forall X. G'_1 \quad G_2 \rightarrow \forall X. G'_2}}{\Omega \vdash t_1 [F] : G'_1[F/X] \setminus X \sqsubseteq t_2 [F] : G'_2[F/X] \setminus X}
\end{array}$$

Figure B.8: $F^?$: Strict type precision and term precision (fragment).