Science of Computer Programming 197 (2020) 102496

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Abstracting gradual references [☆]

Matías Toro [*], Éric Tanter [*]

PLEIAD Laboratory, Computer Science Department (DCC), University of Chile, Santiago, Chile

A R T I C L E   I N F O

Article history:
Received 11 June 2019
Received in revised form 20 May 2020
Accepted 22 May 2020
Available online 8 June 2020

Keywords:
Gradual typing
Mutable references
Abstract interpretation

A B S T R A C T

Gradual typing is an effective approach to integrate static and dynamic typing, which supports the smooth transition between both extremes via the imprecision of type annotations. Gradual typing has been applied in many scenarios such as objects, subtyping, effects, ownership, typestates, information-flow typing, parametric polymorphism, etc. In particular, the combination of gradual typing and mutable references has been explored by different authors, giving rise to four different semantics—invariant, guarded, monotonic and permissive references. These semantics were specially crafted to reflect different design decisions with respect to precision and efficiency tradeoffs. Since then, progress has been made in the formulation of methodologies to *systematically* derive gradual counterparts of statically-typed languages, but these have not been applied to study mutable references.

In this article, we explore how the Abstracting Gradual Typing (AGT) methodology, which has been shown to be effective in a variety of settings, applies to mutable references. Starting from a standard statically-typed language with references, we systematically derive with AGT a novel gradual language, called $\lambda_{\widetilde{\text{REF}}}$. We establish the properties of $\lambda_{\widetilde{\text{REF}}}$; in particular, it is the first gradual language with mutable references that is proven to satisfy the gradual guarantee. We then compare $\lambda_{\widetilde{\text{REF}}}$ with the main four existing approaches to gradual references, and show that the application of AGT does justify one of the proposed semantics: we formally prove that the treatment of references in $\lambda_{\widetilde{\text{REF}}}$ corresponds to the guarded semantics, by presenting a bisimilation with the coercion semantics of Herman et al. In the process, we uncover that any direct application of AGT yields a gradual language that is not space-efficient. We consequently adjust the dynamic semantics of $\lambda_{\widetilde{\text{REF}}}$ to recover space efficiency. We then show how to extend $\lambda_{\widetilde{\text{REF}}}$ to support both monotonic and permissive references as well. Finally, we provide the first proof of the dynamic gradual guarantee for monotonic references. As a result, this paper sheds further light on the design space of gradual languages with mutable references and contributes to deepening the understanding of the AGT methodology.

1. Introduction

Gradual typing supports the smooth transition between static and dynamic checking based on the (programmer-controlled) precision of type annotations [35,38]. Gradual typing relates types of different precision using consistent type relations, such as *type consistency* (resp. *consistent subtyping*), the gradual counterpart of type equality (resp. subtyping). This

☆ This work is partially funded by FONDECYT Projects 1190058 and 3200583.

* Corresponding authors.
E-mail addresses: mtoro@dcc.uchile.cl (M. Toro), etanter@dcc.uchile.cl (É. Tanter).

https://doi.org/10.1016/j.scico.2020.102496
0167-6423/© 2020 Elsevier B.V. All rights reserved.

approach has been applied in a number of settings, such as objects [36], subtyping [36,20], effects [4,5], ownership [34], typestates [47,21], information-flow typing [14,18,41], session types [26], refinements [30], set-theoretic types [8], Hoare logic [3], parametric polymorphism [1,2,28,27,48,42], and references [35,24,39].

In particular, gradual typing for mutable references has seen the elaboration of various possible semantics: *invariant* references [35], *guarded* references [24], *monotonic* references [39], and *permissive* references [39]. Invariant references are a form of references where reference types are invariant with respect to type consistency. Guarded references admit variance thanks to systematic runtime checks on reference reads and writes; the runtime type of an allocated cell never changes during execution. Guarded references have been formulated in a space-efficient coercion calculus, which ensures that gradual programs do not accumulate unbounded pending checks during execution. Monotonic references favor efficiency over flexibility by only allowing reference cells to vary towards more precise types. This allows reference operations in statically-typed regions to safely proceed without any runtime checks. Permissive references are the most flexible approach, in which reference cells can be initialized and updated to any value of any type at any time.

These four developments reflect different design decisions with respect to gradual references: is the reference type constructor variant under consistency? Can the programmer specify a precise bound on the static type of a reference, and hence on the corresponding heap cell type? Can the heap cell type evolve its precision at runtime, and if yes, how? There is obviously no absolute answer to these questions, as they reflect different tradeoffs. This work explores the semantics that results from the application of a *systematic* methodology to gradualize static type systems. Currently we can find in the literature two methodologies to gradualize statically-typed languages: Abstracting Gradual Typing (AGT) [20], and the Gradualizer [10]. In this work, we consider the AGT methodology as it naturally scales to auxiliary structures such as a mutable heap.

The AGT methodology helps to systematically construct gradually-typed languages by using abstract interpretation [12] at the type level. In brief, AGT interprets gradual types as an abstraction of sets of possible static types, formally captured through a Galois connection. The static semantics of a gradual language are then derived by lifting the semantics of a statically-typed language through this connection, and the dynamic semantics follow by Curry-Howard from proof normalization of the type safety argument. The AGT methodology has been shown to be effective in many contexts: records and subtyping [20], type-and-effects [4,5], refinement types [30,45], set-theoretic and union types [8,44], information-flow typing [41], and parametric polymorphism [42]. However, this methodology has never been applied to mutable references in isolation. Although Toro et al. [41] apply AGT to a language with references, they only gradualize *security levels* of types (*e.g.* $\mathsf{Ref}\ \mathsf{Int}_{?}$), not whole types (*e.g.* $\mathsf{Ref}\ ?$ is not supported). In this article we answer the following open questions: Which semantics for gradually-type references follows by systematically applying AGT? Does AGT justify one of the existing approaches, or does it suggest yet another design? Can we recover other semantics for gradual references, if yes, how?

Contributions. This article first reviews the different existing gradual approaches to mutable references (§2). It then presents the semantics for gradual references that is obtained by applying AGT, and how to accommodate the other semantics. More specifically, this work makes the following contributions:

- We present $\lambda_{\widetilde{\text{REF}}}$, a gradual language with support for mutable references (§4). We derive $\lambda_{\widetilde{\text{REF}}}$ by applying the AGT methodology to a simple language with references called λ_{REF} (§3). This is the first application of AGT that focuses on gradually-typed mutable references.
- We prove that $\lambda_{\widetilde{\text{REF}}}$ satisfies the gradual guarantee of Siek et al. [38]. We also present the first formal statement and proof of the conservative extension of the dynamic semantics of the static language [38], for a gradual language derived using AGT (§4.6).
- We prove that the derived language, $\lambda_{\widetilde{\text{REF}}}$, corresponds to the semantics of *guarded references* from HCC (§5). Formally, given a $\lambda_{\widetilde{\text{REF}}}$ term and its compilation to HCC$^+$ (an adapted version of HCC) we prove that both terms are bisimilar, and that consequently they either both terminate, both fail, or both diverge (§5).
- We observe that $\lambda_{\widetilde{\text{REF}}}$ and HCC$^+$ differ in the order of combination of runtime checks. As a result, HCC is space efficient whereas $\lambda_{\widetilde{\text{REF}}}$ is not: we can write programs in $\lambda_{\widetilde{\text{REF}}}$ that may accumulate an unbounded number of checks. We formalize the changes needed in the dynamic semantics of $\lambda_{\widetilde{\text{REF}}}$ to achieve space efficiency (§5.3). This technique to recover space efficiency is in fact independent from mutable references, and is therefore applicable to other gradual languages derived with AGT.
- We formally describe how to support other gradual reference semantics in $\lambda_{\widetilde{\text{REF}}}$ by presenting $\lambda^{\mathsf{pm}}_{\widetilde{\text{REF}}}$, an extension that additionally supports both *permissive* and *monotonic* references (§6). Finally, we prove for the first time that monotonic references satisfy the dynamic gradual guarantee, a non-trivial result that requires careful consideration of updates to the store.

This article is structured as follows: §2 informally introduces the different main approaches to gradual references through examples. Then, §3 presents the fully-static simple language with references called λ_{REF} from which $\lambda_{\widetilde{\text{REF}}}$ is derived. §4 presents $\lambda_{\widetilde{\text{REF}}}$: we start by showing how to derive $\lambda_{\widetilde{\text{REF}}}$ using AGT (§4 and §4.2), then the static and dynamic semantics (§4.3, §4.4 and §4.5), its properties (§4.6), and finally we show $\lambda_{\widetilde{\text{REF}}}$ in action through examples (§4.7). $\lambda_{\widetilde{\text{REF}}}$ is formally compared with HCC in §5: we first present the static and dynamic semantics of HCC$^+$ (§5.1), formally relate both languages using a bisimulation (§5.2), and then present the changes needed in the dynamic semantics of $\lambda_{\widetilde{\text{REF}}}$ to achieve space efficiency (§5.3). §6 describes an extension to $\lambda_{\widetilde{\text{REF}}}$ to encode both permissive and monotonic references, and explains how we prove the dynamic gradual guarantee for monotonic references. Finally, §7 discusses related work, and §8 concludes. Complete definitions and proofs of the main results can be found in the Appendix. Additionally, we implemented $\lambda_{\widetilde{\text{REF}}}$ as an interactive prototype that displays both typing derivations and reduction traces. All the examples mentioned in this paper are readily available in the online prototype available at https://pleiad.cl/grefs.

2. Gradual typing with references

Introducing gradual typing in a language with mutable references raises a number of design and implementation challenges, which have led to different semantics. After a brief review of what mutable references are, we informally present the four approaches to gradual references from the literature. We then formulate the research question that this paper addresses.

2.1. Background: mutable references

Mutable references are memory cells whose content can vary during execution. A language with mutable references typically introduces three basic operations: *allocation* ref t creates a new reference (*i.e.* heap location) initialized with the value of t, *dereferencing* $!t$ returns the value stored in the reference t, and *assignment* $t_1{:=}t_2$ destructively updates the reference t_1 with the value of t_2. For instance:

```
1  let x = ref 4
2  !x
3  x := 10
4  !x
```

Line 1 creates a new reference and returns a new location o pointing to a mutable cell in the store whose content is 4. Line 2 reads 4 from the current stored value of o. Line 3 updates the stored value of o to 10. And finally, line 4 reads again the current stored value of o, which is now 10.

An allocation term ref t has type Ref T where T is the type of the subterm t. Locations o are not part of the source language; they are introduced during reduction. To type locations we use a store typing Σ, a finite map from locations to types, such that o has type T if $\Sigma(o) = T$. One interesting particularity of reference types is that they are *invariant* with respect to subtyping, *i.e.* Ref T_1 <: Ref T_2 if and only if $T_1 = T_2$ [33]. This observation is key in a gradual language when considering the type consistency relation, as illustrated below.

2.2. Background: gradual typing

The most important component of a gradual language is the type (im)precision relation. Type precision $\sqsubseteq$ is a partial order between gradual types, where we say that the gradual type G_1 is less imprecise (or, more precise) than G_2, notation $G_1 \sqsubseteq G_2$, if G_1 *represents* less static types than G_2. In most gradual languages, *imprecision* is introduced by adding the notion of an unknown type ?, which represents any static type whatsoever, i.e. $G \sqsubseteq ?$ for any G. Type precision helps us to define the type consistency relation $\sim$, the gradual counterpart of the equality relation between static types. For instance, any type is consistent with itself, and the ? type is consistent with any gradual type and, vice versa, any gradual type is consistent with the ? type. The static flexibility given by the type consistency relation is backed up dynamically by inserting casts or coercions at the boundaries between types of different precision, ensuring at runtime that no static assumptions are violated. If a static assumption is violated, then a runtime error is raised. Inserting casts may be involved in a setting with higher-order functions. It is in general impossible to immediately check if a function satisfies a particular type. Therefore, in general functions are wrapped in *proxies* that defer the necessary runtime checks on arguments and return values.

2.3. Existing approaches

We now briefly review the four major elaborations of gradual typing with references that have been proposed in the literature.

Invariant references. Siek and Taha [35] include a treatment of references in their original gradual typing work. However, based on the observation that “allowing variance under reference types compromises type safety”, they impose reference types to be invariant with respect to consistency. In other words, $T_1 \sim T_2$ does *not* imply that Ref $T_1 \sim$ Ref T_2. Consider Example 1 below, where :: is a type ascription operator:

```
1  let x = ref (4 :: ?)
2  let y: Ref Int = x   ⟵ type error
3  y := 10
```

Example 1

In this example, 4 :: ? represents an assertion that 4 has type ?. This is accepted by the type system because Int (the type of 4) is *consistent* with the ascribed type ?. The type of the new reference created at line 1 is inferred from the subterm, so `x` has type Ref ?. The program is rejected at line 2 because Ref ? (the type of `x`) is not consistent with Ref Int (the type of `y`) under the invariant semantics.

Guarded references. Herman et al. [24] develop a space-efficient approach to gradual typing based on coercions [23]. Hereafter, we refer to this language as HCC. HCC includes references, albeit with a different semantics from the one proposed by Siek and Taha [35]. In particular, the type system allows consistency variance for reference types, *i.e.* $T_1 \sim T_2$ implies Ref $T_1 \sim$ Ref T_2. The dynamic semantics of the language is given by translation to a language with coercions. Coercions can be normalized in order to avoid accumulation of wrappers that compromise space efficiency. This normalization *eagerly* combine coercions, detecting some errors immediately—e.g. during the reduction of a function—in contrast to lazier approaches that accumulate wrappers and errors are not detected until the casted function is applied. The resulting semantics is called "guarded" because each reference cell assignment (resp. dereference) is guarded with a coercion from the type of the assigned values (resp. expected type of the read value) to/from the actual type of the reference cell. In other words the runtime type of an allocated cell never changes during execution. The approach is intuitively justified by analogy with how first-class functions can be used at different (consistent) types, provided that the appropriate guards check arguments and return values.

Examples 2, 3 and 4 illustrate the use of guarded references:

```
1  let x = ref (4 :: ?)
2  let y: Ref Bool = x
3  !y ← runtime error
```

Example 2

```
1  let x = ref (4 :: ?)
2  let y: Ref Bool = x
3  y := true
4  !y
```

Example 3

```
1  let x = ref 4
2  let y: ? = true
3  x := y ← runtime error
```

Example 4

Example 2 raises a runtime error at line 3 because it is trying to read a Bool where an Int is stored. Example 3 fixes example 2 by updating the location with an actual boolean before the dereference operation. This is possible because the location is created at type Ref ?, meaning that it can store any value of any type (any type is consistent with ?). In Example 4, because the content of the reference is the (unascribed) number 4, the created reference has type Ref Int. A runtime error is raised at line 3 because the coercion injecting `true` from Bool to ? cannot be combined with the coercion projecting `y` from ? to Int.

Monotonic references. Siek et al. [39] propose a design for gradually-typed references called *monotonic references*. The design is driven by efficiency considerations, namely allowing statically-typed code to be compiled with direct memory access instructions—without coercions or wrappers. Like guarded references, monotonic references are variant under consistency. In order to avoid using reference wrappers in statically-typed code, the runtime type of reference cells is allowed to vary but only towards *more precise* types. This monotonicity restriction ensures that direct reference accesses from statically-typed code are safe. Importantly, casts on references are performed directly on the heap.

Examples 3 and 5 illustrate the use of monotonic references:

```
1  let x = ref (4 :: ?)
2  let y: Ref Bool = x ← runtime error
3  y := true
4  !y
```

Example 3

```
1  let x = ref (4 :: ?)
2  let y: Ref Int = x
3  x := true ← runtime error
```

Example 5

In example 3, when variable x is cast to Ref Bool at line 2, the cast is performed directly on the heap: the cast fails as the stored value has type Int instead of Bool. In example 5, when variable x is cast to Ref Int, the runtime type of the heap cell is updated to the more precise type Int. Therefore, the subsequent assignment of true to x triggers a runtime error at line 3 because Bool is not consistent with Int. Note that under the guarded semantics this program runs without errors. The difference is that accesses to y in the monotonic semantics ensures that the value on the heap is of type Int, while under the guarded semantics a coercion is necessary. For instance, consider changing example 5 at line 3, with a dereference (`!y`). While both semantics yield the same result, operationally there is an important difference: in the guarded semantics, because the content type of the reference is ? but y has type Ref Int, an additional coercion of the underlying value to Int occurs. With monotonic references this coercion is not needed, because the semantics enforce that the stored value is of type Int as soon as the alias y is created.

Permissive references. The monotonicity discipline favors efficiency over flexibility. Siek et al. [39] also develop a flexible notion of *permissive references* on top of the language with monotonic references. In essence, permissive references consist in treating the type of all heap cells as ?. A source-level translation then adds the necessary ascriptions on dereferences and assignments. Note that the transformation would work equivalently using the guarded semantics as target (but not with the invariant semantics).

The following examples present a variation of example 3 using permissive references (left),[1] and the program once transformed to the monotonic language (right):

[1] As in [39], we use ref* t to denote the permissive reference constructor, and Ref* T to denote a permissive reference type.

```
let x = ref* 4
let y: Ref* Bool = x
x := true
!y
```

Example 6

$\rightsquigarrow$

```
let x = ref (4 :: ?)
let y = x
x := (true :: ?)
(!y) :: Bool
```

Example 7

With the permissive semantics, the program does not produce any runtime error. The first line of the transformed program creates a new reference of type Ref ?. The third line shows that every assigned value is first ascribed to the ? type. Therefore the runtime type of the heap cell does not change: it stays ?. Finally, in the last line, since the variable y originally had type Ref* Bool, the dereference value is ascribed to Bool.

Note that the permissive semantics is even more flexible than the guarded semantics: guarded semantics allows programmers to fix the type of heap cells at more precise types than ?.

2.4. *Gradual references, systematically*

These four developments reflect different design decisions with respect to gradual references: is the reference type constructor variant under consistency? Can the programmer specify a precise bound on the static type of a reference, and hence on the corresponding heap cell type? Can the heap cell type evolve its precision at runtime, and if yes, how? Although there is no correct answer to any of these questions, in this article we try to answer them for any gradual language derived systematically with AGT. In particular, we answer the following questions: Which semantics for gradually-type references follows by systematically applying AGT? Does AGT justify one of the existing approaches, or does it suggest yet another design? Can we recover other semantics for gradual references, if yes, how?

In the next sections we proceed as follows. First we present λ_{REF}, a standard statically-typed language with references (§3). Second, we systematically apply AGT to λ_{REF} (§4) and observe the resulting semantics, which we called $\lambda_{\widetilde{\text{REF}}}$ (§2). We observe that $\lambda_{\widetilde{\text{REF}}}$ manifests the guarded references semantics of HCC. Third, we formalize this observation by relating $\lambda_{\widetilde{\text{REF}}}$ with HCC (§5). We present an extension of HCC, called HCC^{+}, and a type-driven translation from $\lambda_{\widetilde{\text{REF}}}$ to HCC^{+}. We prove that a $\lambda_{\widetilde{\text{REF}}}$ term and its translation to HCC^{+} are bisimilar. Fourth, we show that, contrary to HCC^{+}, $\lambda_{\widetilde{\text{REF}}}$ is not space-efficient. We then present the changes needed in the dynamic semantics to recover space efficiency (§5). Finally, we present $\lambda_{\widetilde{\text{REF}}}^{\text{pm}}$, an extension of $\lambda_{\widetilde{\text{REF}}}$ to support other semantics both permissive and monotonic references (§6).

3. Preliminary: the static language λ_{REF}

We now apply AGT to a simple language with references, called λ_{REF}, whose static and dynamic semantics are defined in Figs. 1 and 2, respectively.

Static semantics. The definition of λ_{REF} is standard. We use the metavariable l to range over a countably infinite set Loc of locations. A store typing Σ is a partial function from locations to types. A term t can be a lambda abstraction, a constant b, a variable, an application, a binary operation on constants $\oplus$, a conditional expression, a type ascription, a reference, a dereference, an assignment, or a location. Types may be base types (we use B to abstract over all base types), functions, and references. Ref T represents a reference to a value of type T.

To prepare for the application of AGT, the presentation of the type system follows the convention of Garcia et al. [20], in which the type of each sub-expression is kept opaque, the type relations are made explicit as side conditions, and (partial) type functions are used explicitly instead of relying on matching metavariables. In particular, the *dom* (resp. *cod*) partial function is used to obtain the domain (resp. codomain) of a function type; it is undefined otherwise. We similarly introduce the *tref* partial function to extract the underlying type of a reference type. Save for the use of *tref*, rules (Tref), (Tderef), (Tasgn) and (Tl) are all standard [33]. Type ascriptions are also standard [33], though one could argue they are not essential to a static type system; their essential role will become clearer when turning to the gradual language, as ascriptions allow programmers to control (im)precision and play a key role in the dynamic semantics [20]. We use the θ metafunction to determine the type of constants (*e.g.* $\theta(\text{true}) = \text{Bool}$, $\theta(1) = \text{Int}$).

Dynamic semantics. The dynamic semantics of λ_{REF} is presented in Fig. 2, and is standard as well. The semantics is straightforward using evaluation contexts to reduce terms. A store μ maps locations o to values v. Here $\mu[o \mapsto v]$ stands for a new store in which the location o is mapped to the value v. The domain of a store μ, written $dom(\mu)$, is the set of locations for which the finite map is defined. The expression ref t is evaluated by reducing the term t to a value v, obtaining a fresh location in memory and storing the value at that location. The result of ref v is the newly created location. A dereference expression $!t$ first evaluates the term t to a location o, then returns the value stored in memory at location o. An assignment $t_1 := t_2$ evaluates term t_1 to a location o and evaluates term t_2 to a value v. The expression $o := v$ updates the store at location o with the new value v, and returns unit.

Properties. Type safety of λ_{REF} is established as usual: a well-typed closed term is either a value or it can take a step (along a well typed store) to a term of the same type (and a well-typed store that extends the original one).

$$T \in \textsc{Type},\quad B \in \textsc{BaseType},\quad x \in \textsc{Var},\quad o \in \textsc{Loc},\quad b, \in \textsc{Const},$$
$$t \in \textsc{Term},\quad \oplus \in \textsc{Operator},\quad \Gamma \in \textsc{Var} \overset{\text{fin}}{\rightharpoonup} \textsc{Type},\quad \Sigma \in \textsc{Loc} \overset{\text{fin}}{\rightharpoonup} \textsc{Type}$$

$$\begin{array}{rcll} T & ::= & B \mid T \to T \mid \mathsf{Ref}\ T & \text{(types)} \\ t & ::= & v \mid x \mid t\ t \mid t \oplus t \mid \mathsf{if}\ t\ \mathsf{then}\ t\ \mathsf{else}\ t \mid t :: T \mid \mathsf{ref}\ t \mid !t \mid t{:=}t & \text{(terms)} \end{array}$$

$$(\mathrm{Tx})\dfrac{x : T \in \Gamma}{\Gamma; \Sigma \vdash_s x : T} \qquad (\mathrm{Tc})\dfrac{\theta(b) = B}{\Gamma; \Sigma \vdash_s b : B} \qquad (\mathrm{Tapp})\dfrac{\Gamma; \Sigma \vdash_s t_1 : T_1 \quad \Gamma; \Sigma \vdash_s t_2 : T_2 \quad T_2 = dom(T_1)}{\Gamma; \Sigma \vdash_s t_1\ t_2 : cod(T_1)}$$

$$(\mathrm{Top})\dfrac{\begin{array}{cc}\Gamma; \Sigma \vdash_s t_1 : T_1 & \Gamma; \Sigma \vdash_s t_2 : T_2 \\ ty(\oplus) = B_1 \times B_2 \to B_3 & T_1 = B_1 \quad T_2 = B_2\end{array}}{\Gamma; \Sigma \vdash_s t_1 \oplus t_2 : B_3} \qquad (\mathrm{Tif})\dfrac{\begin{array}{cc}\Gamma; \Sigma \vdash_s t_1 : T_1 & \Gamma; \Sigma \vdash_s t_2 : T_2 \\ \Gamma; \Sigma \vdash_s t_3 : T_3 & T_1 = \mathsf{Bool}\end{array}}{\Gamma; \Sigma \vdash_s \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_1 : equate(T_2, T_3)}$$

$$(\mathrm{T}\lambda)\dfrac{\Gamma, x : T_1; \Sigma \vdash_s t : T_2}{\Gamma; \Sigma \vdash_s (\lambda x : T_1.t) : T_1 \to T_2} \qquad (\mathrm{T}{::})\dfrac{\Gamma; \Sigma \vdash_s t : T' \quad T' = T}{\Gamma; \Sigma \vdash_s (t :: T) : T} \qquad (\mathrm{Tref})\dfrac{\Gamma; \Sigma \vdash_s t : T}{\Gamma; \Sigma \vdash_s \mathsf{ref}\ t : \mathsf{Ref}\ T}$$

$$(\mathrm{Tderef})\dfrac{\Gamma; \Sigma \vdash_s t : T}{\Gamma; \Sigma \vdash_s !t : tref(T)} \qquad (\mathrm{Tasgn})\dfrac{\Gamma; \Sigma \vdash_s t_1 : T_1 \quad \Gamma; \Sigma \vdash_s t_2 : T_2 \quad T_2 = tref(T_1)}{\Gamma; \Sigma \vdash_s t_1 := t_2 : \mathsf{Unit}}$$

$$(\mathrm{To})\dfrac{o : T \in \Sigma}{\Gamma; \Sigma \vdash_s o : \mathsf{Ref}\ T}$$

$$\boxed{T = T}$$

$$\dfrac{}{B = B} \qquad \dfrac{T_1' = T_1 \quad T_2 = T_2'}{T_1 \to T_2 = T_1' \to T_2'} \qquad \dfrac{T_1 = T_2}{\mathsf{Ref}\ T_1 = \mathsf{Ref}\ T_2}$$

$$\begin{array}{ll} dom : \textsc{Type} \rightharpoonup \textsc{Type} & cod : \textsc{Type} \rightharpoonup \textsc{Type} \\ dom(T_1 \to T_2) = T_1 & cod(T_1 \to T_2) = T_2 \\ dom(T)\ \text{undef. otherwise} & cod(T)\ \text{undef. otherwise} \end{array}$$

$$\begin{array}{ll} tref : \textsc{Type} \rightharpoonup \textsc{Type} & equate : \textsc{Type} \rightharpoonup \textsc{Type} \\ tref(\mathsf{Ref}\ T) = T & equate(T, T) = T \\ tref(T)\ \text{undef. otherwise} & equate(T_1, T_2)\ \text{undef. otherwise} \end{array}$$

Fig. 1. λ_{REF}: Syntax and type system.

Proposition 1 *(Type safety). Let* $\varnothing; \Sigma \vdash t : T$. *Then one of the following is true:*

1. t *is a value* v*;*
2. *if* $\Sigma \vdash \mu$ *then* $t \mid \mu \longmapsto t' \mid \mu'$, *where* $\varnothing; \Sigma' \vdash t' : T$ *and* $\Sigma' \vdash \mu'$ *some* $\Sigma' \supseteq \Sigma$.

Proof. The proof is standard and follows from progress and preservation [33]. □

4. Gradualizing λ_{REF}

Once we have defined the static language λ_{REF}, the AGT methodology drives the derivation of its gradual counterpart, $\lambda_{\widetilde{\text{REF}}}$, following three steps:

1. Define the syntax of gradual types and give them meaning by concretization to sets of static types; consequently obtain the most precise abstraction, establishing a Galois connection.
2. Derive the gradual type system by using lifted type predicates and type functions in the typing rules.
3. Derive the runtime semantics of the gradual language by proof normalization of gradual typing derivations.

4.1. Syntax and meaning of gradual types

We start by defining the syntax of gradual types. We decide to allow references to gradual types:

$$\begin{array}{l} G \in \textsc{GType} \\ G ::= B \mid G \to G \mid \mathsf{Ref}\ G \mid ? \quad \text{(gradual types)} \end{array}$$

Terms t are lifted to gradual terms $t \in \textsc{GTerm}$, *i.e.* terms with gradual type annotations.

We then give meaning to gradual types via a concretization function γ from gradual types to non-empty sets of static types. We write $\mathscr{P}^*(\textsc{Type})$ to denote the *non-empty* power set of types. We start from the concretization function for GTFL

$$\begin{array}{lcll}
v & ::= & c \mid \lambda x.t \mid o & \text{(values)} \\
E & ::= & \square \mid E \oplus t \mid v \oplus E \mid E\ t \mid v\ E \mid \mathsf{if}\ E\ \mathsf{then}\ t\ \mathsf{else}\ t \mid \mathsf{ref}\ E \mid !E \mid E{:=}t \mid v{:=}E \mid E :: T & \text{(Contexts)} \\
\mu & := & \mu, o \mapsto v & \text{(store)}
\end{array}$$

$\boxed{t \mid \mu \longrightarrow_s t \mid \mu}$ **Notions of Reduction**

$$b_1 \oplus b_2 \mid \mu \longrightarrow_s b_3 \mid \mu \text{ where } c_3 = b_1 [\![\oplus]\!] b_2 \qquad (\lambda x.t)\ v \mid \mu \longrightarrow_s ([v/x]t) \mid \mu$$

$$\mathsf{if}\ b\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \mid \mu \longrightarrow_s \begin{cases} t_1 \mid \mu & \text{if } b = \mathsf{true} \\ t_2 \mid \mu & \text{if } b = \mathsf{false} \end{cases} \qquad \mathsf{ref}\ \ v \mid \mu \longrightarrow_s o \mid \mu[o \mapsto v] \text{ where } o \notin dom(\mu)$$

$$!o \mid \mu \longrightarrow_s v \mid \mu \text{ where } v = \mu(o) \qquad o{:=}v \mid \mu \longrightarrow_s \mathsf{unit} \mid \mu[o \mapsto v] \qquad v :: T \mid \mu \longrightarrow_s v \mid \mu$$

$\boxed{t \mid \mu \longmapsto_s t \mid \mu}$ **Reduction**

$$\frac{t_1 \mid \mu \longrightarrow_s t_2 \mid \mu'}{E[t_1] \mid \mu \longmapsto_s E[t_2] \mid \mu'}$$

Fig. 2. λ_{REF}: Dynamic semantics.

given by Garcia et al. [20], adding an extra case to deal with reference types. This is the natural lifting of concretization to the reference type constructor: Ref G denotes the set of reference types Ref T for each T in the concretization of G:

Definition 1 *(Concretization).* Let $\gamma : \text{GT}\textsc{ype} \to \mathcal{P}^*(\text{T}\textsc{ype})$ be defined as follows:

$$\begin{aligned}
\gamma(B) &= \{\, B \,\} \\
\gamma(G_1 \to G_2) &= \{\, T_1 \to T_2 \mid T_1 \in \gamma(G_1) \wedge T_2 \in \gamma(G_2) \,\} \\
\gamma(\mathsf{Ref}\ G) &= \{\, \mathsf{Ref}\ T \mid T \in \gamma(G) \,\} \\
\gamma(?) &= \text{T}\textsc{ype}
\end{aligned}$$

The notion of type precision between gradual types coincides with set inclusion of their concretizations:

Definition 2 *(Type Precision).* $G_1 \sqsubseteq G_2$ if and only if $\gamma(G_1) \subseteq \gamma(G_2)$.

Proposition 2 *(Precision, inductively). The following inductive definition of type precision is equivalent to Definition 2.*

$$\frac{}{B \sqsubseteq B} \qquad \frac{G_1 \sqsubseteq G_1' \quad G_2 \sqsubseteq G_2'}{G_1 \to G_2 \sqsubseteq G_1' \to G_2'} \qquad \frac{G_1 \sqsubseteq G_2}{\mathsf{Ref}\ G_1 \sqsubseteq \mathsf{Ref}\ G_2} \qquad \frac{}{G \sqsubseteq ?}$$

Once γ is defined, we proceed to define its corresponding abstraction function:

Definition 3 *(Abstraction).* Let the abstraction function $\alpha : \mathcal{P}^*(\text{T}\textsc{ype}) \to \text{GT}\textsc{ype}$ be defined as follows:

$$\begin{aligned}
\alpha(\{\, B \,\}) &= B \\
\alpha(\{\, \overline{T_{i1} \to T_{i2}} \,\}) &= \alpha(\{\, \overline{T_{i1}} \,\}) \to \alpha(\{\, \overline{T_{i2}} \,\}) \\
\alpha(\{\, \overline{\mathsf{Ref}\ T_i} \,\}) &= \mathsf{Ref}\ \alpha(\{\, \overline{T_i} \,\}) \\
\alpha(\widehat{T}) &= ?\ \text{otherwise}
\end{aligned}$$

The abstraction function preserves type constructors and falls back on the unknown type whenever a heterogeneous set is abstracted. As expected, abstraction preserves the Ref type constructor when all static types in the set are reference types. This abstraction function is both sound and optimal: it produces the *most precise* gradual type that over-approximates a given set of static types.

Proposition 3 *(Galois connection). $\langle \gamma, \alpha \rangle$ is a Galois connection,* i.e.*:*

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

Soundness (a) means that α always produces a gradual type whose concretization over-approximates the original set. Optimality (b) means that α always yields the best (*i.e.* least) sound approximation that gradual types can represent.

4.2. Lifting the type system

In order to obtain the static semantics of $\lambda_{\widetilde{\text{REF}}}$, we lift type relations (here, equality) and type functions (*dom*, *cod*, *tref*, *equate*). Following AGT [20], this lifting is obtained by exploiting the Galois connection we have just established through *existential* lifting.

Definition 4 *(Consistency).* $G_1 \sim G_2$ if and only if $T_1 = T_2$ for some $(T_1, T_2) \in \gamma(G_1) \times \gamma(G_2)$. Inductively:

$$\frac{}{G \sim ?} \qquad \frac{}{? \sim G} \qquad \frac{}{G \sim G} \qquad \frac{G_{21} \sim G_{11} \quad G_{12} \sim G_{22}}{G_{11} \to G_{12} \sim G_{21} \to G_{22}} \qquad \frac{G_1 \sim G_2}{\mathsf{Ref}\ G_1 \sim \mathsf{Ref}\ G_2}$$

As a first result, the concretization function justifies consistency variance for reference types—as adopted by all gradual reference systems, except the invariant semantics of Siek and Taha [35].

Lifting type functions follows abstract interpretation as well. For example, consider a partial function $F : \text{Type} \times \text{Type} \rightharpoonup \text{Type}$. The lifting of F, called $\widetilde{F}$, is defined as $\widetilde{F}(G_1, G_2) = \alpha(\widehat{F}(\gamma(G_1), \gamma(G_2)).$[2] Note that as F is partial, the collecting application of F may be the empty set, which is not part of the domain of α; this situation captures the notion of type errors [20].

For instance, the lifting of the *equate* operator presented in Fig. 1 is defined as follows:

Definition 5. $\widetilde{equate}(G_1, G_2) = \alpha(\{equate(T_1, T_2) \mid (T_1, T_2) \in \gamma(G_1) \times \gamma(G_2)\})$ and it comes as no surprise that this definition coincides with the meet operator in the precision order [20]:

Proposition 4. $\widetilde{equate}(G_1, G_2) = G_1 \sqcap G_2$.

The meet operator is defined as $G_1 \sqcap G_2 = \alpha(\gamma(G_1) \cap \gamma(G_2))$, *and inductively as:*

$$B \sqcap B = B \qquad G_1 \sqcap G_2 = G_2 \sqcap G_1 \qquad G \sqcap ? = ? \sqcap G = G \qquad (G_{11} \to G_{12}) \sqcap (G_{21} \to G_{22}) = (G_{11} \sqcap G_{21}) \to (G_{12} \sqcap G_{22})$$

$$\mathsf{Ref}\ G_1 \sqcap \mathsf{Ref}\ G_2 = \mathsf{Ref}\ G_1 \sqcap G_2 \qquad G_1 \sqcap G_2 \textit{ is undefined otherwise}$$

Compositional lifting. As previously noted by Garcia et al. [20] we cannot always apply compositionally lifting to predicates that use both type relations and type functions. However, we justify that we can do it for application and assignment rules.

Proposition 5. *Let* $P_1(T_1, T_2) \triangleq T_1 = dom(T_2)$. *Then* $\widetilde{P_1}(G_1, G_2) \iff G_1 \sim \widetilde{dom}(G_2)$.

Proposition 6. *Let* $P_2(T_1, T_2) \triangleq T_1 = tref(T_2)$. *Then* $\widetilde{P_2}(G_1, G_2) \iff G_1 \sim \widetilde{tref}(G_2)$.

Consistent reference type function. The algorithmic consistent lifting of the *tref* type function, $\widetilde{tref}$, is provided in Fig. 3. As expected, it justifies the fact that a term of the unknown type $?$ can be dealt with as if it were a reference type $\mathsf{Ref}\ ?$, since $\widetilde{tref}(?) = ?$.

4.3. Static semantics

The type system of $\lambda_{\widetilde{\text{REF}}}$ is presented in Fig. 3, along algorithmic definitions of consistent functions; the type rules are obtained by replacing type predicates and functions with their corresponding liftings. For simplicity, we use notation $t : G$ if $\cdot \vdash t : G$. Ascriptions play an important role in the gradual language [20]: they conveniently allow programmers to introduce (im)precision as desired. For instance, the following program typechecks due to the convenient use of ascriptions: $((\lambda b : \mathsf{Bool}.\mathsf{if}\ b\ \mathsf{then}\ \mathsf{true} :: ?\ \mathsf{else}\ 1 :: ?)\ \mathsf{false}) + 2$. Note that the type equality premise of ascription in λ_{REF} is lifted to a type consistency premise in $\lambda_{\widetilde{\text{REF}}}$. As we will see in the next section, ascriptions are also helpful in the dynamic semantics to ensure that type precision is stable under substitution, hence ensuring typing preservation.

4.4. Dynamic semantics

Traditionally, the dynamic semantics of a gradual language is given by translation to an intermediate cast calculus [35]. One of the salient features of the AGT methodology is that it provides a *direct* dynamic semantics of gradual programs, defined over gradual typing derivations [20]. The key idea is to apply *proof reduction* on gradual typing derivations [25]

[2] $\widehat{F}$ is notation for the collecting function of F.

$$G \in \text{GType}, \quad B \in \text{BaseType}, \quad x \in \text{Var}, \quad o \in \text{Loc}, \quad b \in \text{Const},$$

$$t \in \text{GTerm}, \quad \oplus \in \text{Operator}, \quad \Gamma \in \text{Var} \overset{\text{fin}}{\rightharpoonup} \text{GType}, \quad \Sigma \in \text{Loc} \overset{\text{fin}}{\rightharpoonup} \text{GType}$$

$$\begin{array}{lll} G & ::= \; ? \mid B \mid G \to G \mid \mathsf{Ref}\ G & \text{(types)} \\ t & ::= \; b \mid (\lambda x : G_1.t) \mid o \mid x \mid t\ t \mid t \oplus t \mid \mathsf{if}\ t\ \mathsf{then}\ t\ \mathsf{else}\ t \mid t :: T \mid \mathsf{ref}\ t \mid !t \mid t{:=}t & \text{(terms)} \end{array}$$

$$(Gx)\frac{x : G \in \Gamma}{\Gamma;\Sigma \vdash x : G} \qquad (Gc)\frac{\theta(b) = B}{\Gamma;\Sigma \vdash b : B} \qquad (Gapp)\frac{\Gamma;\Sigma \vdash t_1 : G_1 \quad \Gamma;\Sigma \vdash t_2 : G_2 \quad G_2 \sim \widetilde{dom}(G_1)}{\Gamma;\Sigma \vdash t_1\ t_2 : \widetilde{cod}(G_1)}$$

$$(Gop)\frac{\begin{array}{cc}\Gamma;\Sigma \vdash t_1 : G_1 & \Gamma;\Sigma \vdash t_2 : G_2 \\ ty(\oplus) = B_1 \times B_2 \to B_3 & G_1 \sim B_1 \quad G_2 \sim B_2\end{array}}{\Gamma;\Sigma \vdash t_1 \oplus t_2 : B_3} \qquad (Gif)\frac{\begin{array}{cc}\Gamma;\Sigma \vdash t_1 : G_1 & \Gamma;\Sigma \vdash t_2 : G_2 \\ \Gamma;\Sigma \vdash t_3 : G_3 & G_1 \sim \mathsf{Bool}\end{array}}{\Gamma;\Sigma \vdash \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 : G_2 \sqcap G_3}$$

$$(G\lambda)\frac{\Gamma, x : G_1 \vdash t : G_2}{\Gamma;\Sigma \vdash (\lambda x : G_1.t) : G_1 \to G_2} \qquad (G{::})\frac{\Gamma;\Sigma \vdash t : G' \quad G' \sim G}{\Gamma;\Sigma \vdash (t :: G) : G} \qquad (Gref)\frac{\Gamma;\Sigma \vdash t : G}{\Gamma;\Sigma \vdash \mathsf{ref}\ t : \mathsf{Ref}\ G}$$

$$(Gderef)\frac{\Gamma;\Sigma \vdash t : G}{\Gamma;\Sigma \vdash !t : \widetilde{tref}(G)} \qquad (Gasgn)\frac{\Gamma;\Sigma \vdash t_1 : G_1 \quad \Gamma;\Sigma \vdash t_2 : G_2 \quad G_2 \sim \widetilde{tref}(G_1)}{\Gamma;\Sigma \vdash t_1 := t_2 : \mathsf{Unit}}$$

$$(Go)\frac{o : G \in \Sigma}{\Gamma;\Sigma \vdash o : \mathsf{Ref}\ G}$$

$$\begin{array}{lll} \widetilde{dom} : \text{GType} \rightharpoonup \text{GType} & \widetilde{cod} : \text{GType} \rightharpoonup \text{GType} & \widetilde{tref} : \text{GType} \rightharpoonup \text{GType} \\ \widetilde{dom}(G_1 \to G_2) = G_1 & \widetilde{cod}(G_1 \to G_2) = G_2 & \widetilde{tref}(\mathsf{Ref}\ G) = G \\ \widetilde{dom}(?) = ? & \widetilde{cod}(?) = ? & \widetilde{tref}(?) = ? \\ \widetilde{dom}(G) \text{ undefined otherwise} & \widetilde{cod}(G) \text{ undefined otherwise} & \widetilde{tref}(G) \text{ undefined otherwise} \end{array}$$

Fig. 3. $\lambda_{\widetilde{\text{REF}}}$: Syntax and typing rules.

augmented with *evidence* for consistent judgments. By the Curry-Howard correspondence, this induces a notion of reduction for gradual terms.

More specifically, the reduction of gradual typing derivations mirrors reasoning steps used in the type safety proof of the static language. The static type safety proof relies on transitivity of type relations, but in a gradual setting, transitivity does not always hold. For instance, equality is a transitive type relation, but type consistency—which only captures plausibility—is not transitive in general: $\mathsf{Int} \sim ?$ and $? \sim \mathsf{Bool}$, but $\mathsf{Int} \nsim \mathsf{Bool}$. In AGT, gradual typing derivations are augmented with type-based justifications of *why* a consistent judgment holds, called *evidence*. Evidence can generally be represented by a pair of gradual types, $\varepsilon = \langle G_1, G_2 \rangle$, which capture the implied information about types related by a consistent judgment; these types are at least as precise as the types involved in the judgment [20]. We use notation $\varepsilon \vdash G_1 \sim G_2$ to denote that evidence ε justifies the consistency judgment $G_1 \sim G_2$. During proof reduction (which corresponds to a reduction step), when a transitivity claim between two consistent judgments needs to be justified, the corresponding evidences of these judgments are combined via *consistent transitivity*. If the combination is defined, then the resulting evidence justifies the new consistent judgment and the reduction step can be taken, otherwise the program halts with an error.

Evidence is initially computed by a partial function called an *initial evidence operator* $\mathcal{I}_=$ [20]. An initial evidence operator computes the most precise evidence that can be deduced from a given judgment. For instance the initial evidence of consistent judgment $G_1 \sim G_2$ is $\varepsilon = \mathcal{I}_=(G_1, G_2)$, i.e. $\mathcal{I}_=(G_1, G_2) \vdash G_1 \sim G_2$. Formally the initial evidence operator is defined as:

Definition 6 *(Initial evidence operator).*

$$\mathcal{I}_=(G_1, G_2) = \alpha^2(\{\langle T_1, T_2 \rangle \mid T_1 \in \gamma(G_1), T_2 \in \gamma(G_2), T_1 = T_2)\})$$

Given two sets of static types that belong to the concretization of both gradual types, this function abstracts the sets of pairs of static types such that both types are equal.[3] In this setting with only consistency (and not consistent subtyping), the initial evidence operator coincides with the pair of meets between the two types.

Proposition 7. *If* $G_1 \sim G_2$, *then* $\mathcal{I}_=(G_1, G_2) = \langle G_1 \sqcap G_2, G_1 \sqcap G_2 \rangle$.

[3] $\alpha^2(\{\langle \overline{T_{i1}, T_{i2}} \rangle\}) = \langle \alpha(\{\overline{T_{i1}}\}), \alpha(\{\overline{T_{i2}}\}) \rangle$.

Illustration. Consider the gradual typing derivation of $(\lambda x : ?. x + 1)\ \mathsf{false}$, called $\mathcal{D}$ (for simplicity we omit the store typing):

$$\mathcal{D} = \frac{\mathcal{D}' \qquad \dfrac{\theta(\mathsf{false}) = \mathsf{Bool}}{\varnothing \vdash \mathsf{false} : \mathsf{Bool}} \qquad \langle \mathsf{Bool}, \mathsf{Bool}\rangle \vdash \mathsf{Bool} \sim ?}{\varnothing \vdash (\lambda x : ?. x + 1)\ \mathsf{false} : \mathsf{Int}}$$

$$\text{where } \mathcal{D}' = \frac{\dfrac{\dfrac{x : ? \in x : ?}{x : ? \vdash x : ?} \qquad \dfrac{\theta(1) = \mathsf{Int}}{x : ? \vdash 1 : \mathsf{Int}} \qquad \langle \mathsf{Int}, \mathsf{Int}\rangle \vdash ? \sim \mathsf{Int} \qquad \langle \mathsf{Int}, \mathsf{Int}\rangle \vdash \mathsf{Int} \sim \mathsf{Int}}{x : ? \vdash x + 1}}{\varnothing \vdash (\lambda x : ?. x + 1) : ? \rightarrow \mathsf{Int}}$$

In the typing derivation of the function $\mathcal{D}'$, the consistent judgments $? \sim \mathsf{Int}$ and $\mathsf{Int} \sim \mathsf{Int}$ support the addition expression, and at the top-level, the judgment $\mathsf{Bool} \sim ?$ supports the application of the function to false. By knowing that $? \sim \mathsf{Int}$ 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 $\varepsilon_1 = \mathcal{I}_{=}(?, \mathsf{Int}) = \langle \mathsf{Int}, \mathsf{Int}\rangle$, i.e. $\langle \mathsf{Int}, \mathsf{Int}\rangle \vdash ? \sim \mathsf{Int}$. For the $\mathsf{Int} \sim \mathsf{Int}$ consistent judgment we cannot learn anything new, therefore its evidence is $\varepsilon_2 = \mathcal{I}_{=}(\mathsf{Int}, \mathsf{Int}) = \langle \mathsf{Int}, \mathsf{Int}\rangle$. Similarly, the evidence for the third judgment is $\varepsilon_3 = \mathcal{I}_{=}(\mathsf{Bool}, ?) = \langle \mathsf{Bool}, \mathsf{Bool}\rangle$. □

At runtime, reduction rules need to *combine* evidence in order to either justify or refute a use of transitivity in the type preservation argument. The combination operation, called *consistent transitivity* $\circ^{=}$, determines whether two evidences support the transitivity of their corresponding judgments. The definition of consistent transitivity for a type predicate P, $\circ^{P}$, is given by the abstract interpretation framework [20]; in particular, for type equality it is defined as follows[4]:

Definition 7 *(Consistent transitivity).* 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 $\varepsilon_{ab} \circ^{=} \varepsilon_{bc} \vdash G_a \sim G_b$, where:

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

As $G_1 = G_{21}$ and $G_{22} = G_3$, the definition of consistent transitivity corresponds to the meet of gradual types $\sqcap$:

Lemma 8. $\langle G_1\rangle \circ^{=} \langle G_2\rangle = \langle G_1 \sqcap G_2, G_1 \sqcap G_2\rangle$.

The only operators that create new evidence are the initial evidence and consistent transitivity operators. These two operators always return evidence where both components are the same, therefore for simplicity we use notation $\langle G\rangle$ instead of $\langle G, G\rangle$.

Illustration. Let us go back to the example above. The gradual typing derivation $\mathcal{D}$ is reduced by using preservation arguments as follows:

$$\mathcal{D} \longmapsto \frac{\dfrac{\dfrac{\theta(\mathsf{false}) = \mathsf{Bool}}{\varnothing \vdash \mathsf{false} : \mathsf{Bool}} \qquad \langle \mathsf{Bool}\rangle \vdash \mathsf{Bool} \sim ?}{\varnothing \vdash \mathsf{false} :: ? : ?} \qquad \dfrac{\theta(1) = \mathsf{Int}}{\varnothing \vdash 1 : \mathsf{Int}} \qquad \langle \mathsf{Int}\rangle \vdash ? \sim \mathsf{Int} \qquad \langle \mathsf{Int}\rangle \vdash \mathsf{Int} \sim \mathsf{Int}}{\varnothing \vdash (\mathsf{false} :: ?) + 1 : \mathsf{Int}} \longmapsto \mathbf{error}$$

Note that the use of ascriptions is crucial to represent each step of evaluation as a legal source typing, and most importantly to preserve evidence of different subterms. In this case, instead of replacing x with false, we replace x with $\mathsf{false} :: ?$, otherwise (1) the consistent judgment $\langle \mathsf{Bool}\rangle \vdash \mathsf{Bool} \sim ?$ would be lost, and (2) the resulting gradual typing derivation would be ill-typed. To simplify the new ascription to $?$, we need to combine ε_1 and ε_3 in order to (try to) obtain a justification for the transitive judgment, namely that $\mathsf{Bool} \sim \mathsf{Int}$, but $\varepsilon_3 \circ^{=} \varepsilon_1 = \langle \mathsf{Bool}\rangle \circ^{=} \langle \mathsf{Int}\rangle = \langle \mathsf{Bool} \sqcap \mathsf{Int}\rangle$, which is undefined, so a runtime error is raised. □

To formalize this approach to the runtime semantics of gradual programs while avoiding writing reduction rules on actual (bi-dimensional) derivation trees, Garcia et al. adopt *intrinsic terms* [9], which are a flat notation that is isomorphic to typing derivations. In this paper, we use the same technique, and introduce the semantics via a language of intrinsic terms, called $\lambda^{\varepsilon}_{\widetilde{\mathsf{REF}}}$.

4.4.1. *Static semantics of* $\lambda^{\varepsilon}_{\widetilde{\mathsf{REF}}}$

The syntax and static semantics of $\lambda^{\varepsilon}_{\widetilde{\mathsf{REF}}}$ is presented in Fig. 4. Intrinsically-typed terms t^{G} comprise a family $\mathbb{T}[G]$ of type-indexed sets, such that ill-typed terms do not exist. Intrinsic terms are built up from disjoint families $x^{G} \in \mathbb{V}[G]$ and

[4] The consistent transitivity operator is parametrized by the type predicate being lifted. For instance, in other settings such as for subtyping, we write $\circ^{<:}$ instead, which is defined analogously.

$$
\begin{array}{l}
et \in \textsc{EvTerm}, \quad \langle G\rangle \in \textsc{Evidence}, \quad u \in \textsc{SimpleValue}, \quad v \in \textsc{Value}, \quad t \in \mathbb{T}[*],\\
u \ ::= \ x \mid b \mid \lambda x.t \mid o^{G}\\
v \ ::= \ u \mid \langle G\rangle u :: G\\
et \ ::= \ \langle G\rangle t\\
t \ ::= \ v \mid et \oplus et \mid et \mathbin{@^{G}} et \mid et :: G \mid \mathsf{if}\ et\ \mathsf{then}\ et\ \mathsf{else}\ et \mid \mathsf{ref}^{G}\ et \mid !^{G} et \mid et :=^{G} et
\end{array}
$$

$$
(\text{IGb}\ \dfrac{\theta(b) = B}{b \in \mathbb{T}[B]}
\qquad
(\text{IGx})\dfrac{}{x^{G} \in \mathbb{T}[G]}
\qquad
(\text{IGapp})\dfrac{\begin{array}{c} t^{G_1} \in \mathbb{T}[G_1] \quad t^{G_2} \in \mathbb{T}[G_2] \\ \langle G_1'\rangle \vdash G_1 \sim G_{11} \to G_{12} \\ \langle G_2'\rangle \vdash G_2 \sim G_{11}\end{array}}{(\langle G_1'\rangle t^{G_1}) \mathbin{@^{G_{11}\to G_{12}}} (\langle G_2'\rangle t^{G_2}) \in \mathbb{T}[G_{12}]}
$$

$$
(\text{IG}\oplus)\dfrac{\begin{array}{c} t^{G_1} \in \mathbb{T}[G_1] \quad \langle B_1\rangle \vdash G_1 \sim B_1 \\ t^{G_2} \in \mathbb{T}[G_2] \quad \langle B_2\rangle \vdash G_2 \sim B_2 \\ ty(\oplus) = B_1 \times B_2 \to B_3 \end{array}}{\langle B_1\rangle t^{G_1} \oplus \langle B_2\rangle t^{G_2} \in \mathbb{T}[B_3]}
\qquad
(\text{IGif})\dfrac{\begin{array}{c} G = (G_2 \sqcap G_3) \\ t^{G_1} \in \mathbb{T}[G_1] \quad \langle \mathsf{Bool}\rangle \vdash G_1 \sim \mathsf{Bool} \\ t^{G_2} \in \mathbb{T}[G_2] \quad t^{G_3} \in \mathbb{T}[G_3] \end{array}}{\mathsf{if}\ \langle \mathsf{Bool}\rangle t^{G_1}\ \mathsf{then}\ \langle G\rangle t^{G_2}\ \mathsf{else}\ \langle G\rangle t^{G_3} \in \mathbb{T}[G]}
$$

$$
(\text{IG}\lambda)\dfrac{t^{G_2} \in \mathbb{T}[G_2]}{\lambda x^{G_1}.t^{G_2} \in \mathbb{T}[G_1 \to G_2]}
\qquad
(\text{IG}::)\dfrac{t^{G_1} \in \mathbb{T}[G_1] \quad \langle G_2'\rangle \vdash G_1 \sim G_2}{\langle G_2'\rangle t^{G_1} :: G_2 \in \mathbb{T}[G_2]}
$$

$$
(\text{IGref})\dfrac{t^{G_1} \in \mathbb{T}[G_1] \quad \langle G'\rangle \vdash G_1 \sim G_2}{\mathsf{ref}^{G_2}\ \langle G'\rangle t^{G_1} \in \mathbb{T}[\mathsf{Ref}\ G_2]}
\qquad
(\text{IGderef})\dfrac{t^{G_2} \in \mathbb{T}[G_2] \quad \langle \mathsf{Ref}\ G_1\rangle \vdash G_2 \sim \mathsf{Ref}\ G}{!^{G}(\langle \mathsf{Ref}\ G_1\rangle t^{G_2}) \in \mathbb{T}[G]}
$$

$$
(\text{IGasgn})\dfrac{\begin{array}{c} t^{G_1} \in \mathbb{T}[G_1] \quad \langle G_1'\rangle \vdash G_1 \sim \mathsf{Ref}\ G_3 \\ t^{G_2} \in \mathbb{T}[G_2] \quad \langle G_2'\rangle \vdash G_2 \sim G_3 \end{array}}{\langle G_1'\rangle t^{G_1} :=^{G_3} \langle G_2'\rangle t^{G_2} \in \mathbb{T}[\mathsf{Unit}]}
\qquad
(\text{IGl})\dfrac{}{o^{G} \in \mathbb{T}[\mathsf{Ref}\ G]}
$$

Fig. 4. $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$: Syntax and typing rules.

$o^{G} \in \mathbb{L}[G]$ of intrinsically-typed variables and locations respectively. Note that intrinsic terms do not need explicit type environment Γ or store environments Σ. Essentially, an intrinsic term $t^{G} \in \mathbb{T}[G]$ represents the typing derivation for the $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ judgment $\Gamma; \Sigma \vdash t : G$, where Γ and Σ correspond to the free (intrinsically-typed) variables and locations in t^{G}. We omit the type exponent on intrinsic terms when not needed, writing for instance $t \in \mathbb{T}[G]$.

The syntax and type rules for intrinsic terms is presented in Fig. 4. We use notation et to refer to an *evidence term*, which are terms augmented with evidence. This evidence justifies why the type of the term is consistent with the corresponding statically determined type.[5] For instance, in term $\langle \mathsf{Int}\rangle 1 :: ?$, evidence $\langle \mathsf{Int}\rangle$ is the companion of the raw value 1 and justifies that $\mathsf{Int} \sim ?$. Intrinsic values v can either be simple values u or ascribed values $\varepsilon u :: G$. A simple value u can be a variable x, a constant b, a lambda abstraction $\lambda x.t$, or a location o^{G}. Some terms carry extra type annotations purely to help prove type safety, such as G in $et :=^{G} et$, and to ensure unicity of typing during reduction such as G in $et \mathbin{@^{G}} et$. The type rules mirror the type rules of $\lambda_{\widetilde{\text{REF}}}$ where each consistent judgment is justified by some evidence. The presentation may differ sometimes: for instance in Rule (IGasgn), its extrinsic counterpart has premise $\widetilde{tref}(G_1) \sim G_2$ which is equivalent to both $G_1 \sim \mathsf{Ref}\ G_3$ and $G_2 \sim G_3$. We choose the later representation because it allows us to track evidence for each of the subterms. Something similar occurs in rules (IGderef) and (IGref): extrinsic rules (Gderef) and (Gref) has no consistent judgment whatsoever. This judgment is justified as subterms may evolve during reduction into something of a different (but consistent) type. For instance, in rule (IGderef), evidence $\langle \mathsf{Ref}\ G_1\rangle$ justifies that the type of subterm t^{G_2} is consistent with $\mathsf{Ref}\ G$, the type of the subterm during type checking. Alternatively, rule (IGderef) may also be seen as the intrinsic counterpart of the following $\lambda_{\widetilde{\text{REF}}}$ rule:

$$
(\text{Gderef*})\dfrac{\Gamma; \Sigma \vdash t : G_1 \qquad G_1 \sim \mathsf{Ref}\ G_2}{\Gamma; \Sigma \vdash\ !^{G_2} t : G_2}
$$

where statically $G_1 = \mathsf{Ref}\ G_2$. The elaboration rules for intrinsic terms, i.e. from $\lambda_{\widetilde{\text{REF}}}$ to $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$, are explained later in §4.5.

[5] As illustrated previously, evidence lives in a derivation tree, to justify a consistent judgment. When moving to the flat representation of intrinsic terms, the question arises of where to put the evidence. If a consistent judgment naturally corresponds to a subterm, then we annotate that subterm with evidence. Note however that in some gradual languages, such as security-typed languages, some consistent judgments may not correspond to one subterm, and in that case the practice is to decorate the term constructor itself [41].

4.4.2. Dynamic semantics of $\lambda^{\varepsilon}_{\widetilde{REF}}$

Now we turn to the reduction rules of intrinsic terms, possibly failing with an **error** when combining evidences using consistent transitivity defined above. The reduction rules are presented in Fig. 5. They are defined over configurations $\textsc{Config}_G$ which consist of a pair of a term and a store. Contrary to [20], instead of using evaluation frames, we define the reduction semantics by using an equivalent representation using *evaluation contexts* [17], which make it easier to recover space efficiency (§5.3). We explain how to derive Rules $(r4)$, $(r5)$, and $(r6)$, which deal with references, in §4.4.3.

Rules $(r1)$, $(r2)$, and $(r3)$, present no novelty with respect to the presentation of Garcia et al. [20].

Rule $(r4)$ reduces a reference to a new location o^G not already present in the domain of store μ. The store is extended mapping o^G to the evidence value ascribed to G_2, the type of the reference determined statically. This use of ascriptions to anchor new evidence and preserve typing upon reduction is used in almost all other reduction rules.

Rule $(r5)$ reduces a dereference to the underlying value v of location o^{G_2}, ascribed to the statically determined type G, where evidence $\langle G_1\rangle$ justifies that G_2 (the type of v), is consistent with G.

Rule $(r6)$ updates the corresponding value on the heap of location o^{G_1}, with raw value u ascribed to G. As $\langle G_2\rangle$ justifies that the type of u is consistent with G_3, and by inversion lemmas, $\langle G_1\rangle$ justifies that $G_3 \sim G$, then evidence $\langle G_2\rangle \circ^{=} \langle G_1\rangle = \langle G_2 \sqcap G_1\rangle$ (if defined) justifies that the type of u is consistent with G. If $G_2 \sqcap G_1$ is not defined then a runtime error is signaled.

4.4.3. Deriving the reduction rules of $\lambda^{\varepsilon}_{\widetilde{REF}}$

We now intuitively describe how we derive all reference related rules: $(r4)$, $(r5)$ and $(r6)$.

Rule $(r4)$. We start with the last intrinsic term before elimination

$$(\text{IGref})\,\frac{u \in \mathbb{T}[G_1] \qquad \langle G'\rangle \vdash G_1 \sim G_2}{\mathsf{ref}^{G_2}\ \langle G'\rangle u \in \mathbb{T}[\mathsf{Ref}\ G_2]}$$

By following the reduction rule for allocating a reference for λ_{REF} (Fig. 2), we would have to reduce allocations as follows:

$$\mathsf{ref}^{G_2}\ \langle G'\rangle u \mid \mu \longrightarrow o^{G_2} \mid \mu[o^{G_2} \mapsto u]$$

where $o^{G_2} \notin dom(\mu)$. But u does not have type G_2, therefore we ascribe the raw value to G_2 using some evidence that justifies that $G_1 \sim G_2$. As we already know that $\langle G'\rangle \vdash G_1 \sim G_2$, we can finally derive the reduction rule for allocations as follows:

$$\mathsf{ref}^{G_2}\ \langle G'\rangle u \mid \mu \longrightarrow o^{G_2} \mid \mu[o^{G_2} \mapsto \langle G_2\rangle u :: G_2]$$

Rule $(r5)$. Similarly to $(r4)$, we start from the last intrinsic term before elimination:

$$(\text{IGderef})\,\frac{o^{G_2} \in \mathbb{T}[\mathsf{Ref}\ G_2] \qquad \langle \mathsf{Ref}\ G_1\rangle \vdash \mathsf{Ref}\ G_2 \sim \mathsf{Ref}\ G}{!^{G}(\langle \mathsf{Ref}\ G_1\rangle o^{G_2}) \in \mathbb{T}[G]}$$

Following Fig. 2 we would have to reduce dereferences as follows:

$$!^{G}(\langle \mathsf{Ref}\ G_1\rangle o^{G_2}) \mid \mu \longrightarrow v \mid \mu$$

where $\mu(o^{G_2}) = v$. But as $v \in \mathbb{T}[G_2]$ and the expected type of the dereference is G, we need to ascribe the dereferenced value to G. Now there is the question about what evidence to use. Of course, we cannot make up new evidence from thin air, we have to use and combine evidence already present in the redex. We know from the premise of the redex that $\langle \mathsf{Ref}\ G_1\rangle \vdash \mathsf{Ref}\ G_2 \sim \mathsf{Ref}\ G$, and by an inversion lemma we also know that $\langle G_1\rangle \vdash G_2 \sim G$, which is exactly what we need. The final reduction rule for dereferences is therefore:

$$!^{G}(\langle \mathsf{Ref}\ G_1\rangle o^{G_2}) \mid \mu \longrightarrow \langle G_1\rangle v :: G \mid \mu$$

Rule $(r6)$. Let us start from an assignment intrinsic term before elimination:

$$(\text{IGasgn})\,\frac{\begin{array}{cc} o^{G_1} \in \mathbb{T}[\mathsf{Ref}\ G_1] & \langle \mathsf{Ref}\ G_1'\rangle \vdash \mathsf{Ref}\ G_1 \sim \mathsf{Ref}\ G_3 \\ u \in \mathbb{T}[G_2] & \langle G_2'\rangle \vdash G_2 \sim G_3 \end{array}}{\langle G_1'\rangle o^{G_1} :=^{G_3} \langle G_2'\rangle u \in \mathbb{T}[\mathsf{Unit}]}$$

If we follow the reduction rule for assignment of λ_{REF} (Fig. 2), then we would be tempted to reduce assignments as:

$$\langle G_1'\rangle o^{G_1} :=^{G_3} \langle G_2'\rangle u \mid \mu \longrightarrow \mathsf{unit} \mid \mu[o^{G_1} \mapsto u]$$

$$\begin{array}{lcl}
ev \in \textsc{EvValue}, & t \in \mathbb{T}[*], & F \in \textsc{EvCtx}, \quad E \in \textsc{TmFrame} \\
ev & ::= & \langle G\rangle u \\
F & ::= & \square \mid E \oplus et \mid ev \oplus E \mid E \mathbin{@^{G}} et \mid ev \mathbin{@^{G}} E \mid E :: G \mid \\
 & & \text{if } E \text{ then } et \text{ else } et \mid \text{ref}^{G}\ E \mid !^{G} E \mid E :=^{G} et \mid ev :=^{G} E \\
E & ::= & F \mid \langle G\rangle F \\
\mu & := & \cdot \mid \mu, o^{G} \mapsto v
\end{array}$$

Notions of Reduction

$$\textsc{Config}_G = \mathbb{T}[G] \times \textsc{Store}$$

$$\boxed{\begin{array}{l}\longrightarrow : \textsc{Config}_G \times (\textsc{Config}_G \cup \{\,\textbf{error}\,\}) \\ \longrightarrow_c : \textsc{EvTerm} \times (\textsc{EvTerm} \cup \{\,\textbf{error}\,\})\end{array}}$$

$$(r1) \qquad \langle B_1\rangle b_1 \oplus \langle B_2\rangle b_2 \mid \mu \longrightarrow b_3 \mid \mu \quad \text{where } b_3 = b_1 \llbracket \oplus \rrbracket b_2$$

$$(r2)\ (\langle G'_{11} \to G'_{12}\rangle(\lambda x^{G_{11}}.t)) @^{G_1 \to G_2} (\langle G'_2\rangle u) \mid \mu \longrightarrow \begin{cases} \langle G'_{12}\rangle([(\langle G'_2 \sqcap G'_{11}\rangle u :: G_{11})/x^{G_{11}}]t) :: G_2 \mid \mu \\ \textbf{error} \qquad \text{if } G'_2 \sqcap G'_{11} \text{ is not defined} \end{cases}$$

$$(r3) \qquad \text{if } \langle \text{Bool}\rangle b \text{ then } \langle G\rangle t^{G_2} \text{ else } \langle G\rangle t^{G_3} \mid \mu \longrightarrow \begin{cases} \langle G\rangle t^{G_2} :: G \mid \mu & b = \text{true} \\ \langle G\rangle t^{G_3} :: G \mid \mu & b = \text{false} \end{cases} \quad \text{where } G = G_2 \sqcap G_3$$

$$(r4) \qquad \text{ref}^{G_2}\ \langle G_1\rangle u \mid \mu \longrightarrow o^{G_2} \mid \mu[o^{G_2} \mapsto \langle G_1\rangle u :: G_2] \quad \text{where } o \notin dom(\mu)$$

$$(r5) \qquad !^{G}(\langle \text{Ref } G_1\rangle o^{G_2}) \mid \mu \longrightarrow \langle G_1\rangle v :: G \mid \mu \quad \text{where } v = \mu(o^{G_2})$$

$$(r6) \qquad \langle \text{Ref } G_1\rangle o^{G} :=^{G_3} \langle G_2\rangle u \mid \mu \longrightarrow \begin{cases} \text{unit} \mid \mu[o^{G} \mapsto \langle G_2 \sqcap G_1\rangle u :: G] \\ \textbf{error} \qquad \text{if } G_2 \sqcap G_1 \text{ is not defined} \end{cases}$$

$$(r7) \qquad \langle G_2\rangle(\langle G_1\rangle u :: G) \longrightarrow_c \begin{cases} \langle G_1 \sqcap G_2\rangle u \\ \textbf{error} \qquad \text{if } G_1 \sqcap G_2 \text{ is not defined} \end{cases}$$

$$\boxed{\longmapsto : \textsc{Config}_G \times (\textsc{Config}_G \cup \{\,\textbf{error}\,\})} \ \textbf{Reduction}$$

$$(\text{R}E)\ \frac{t_1^{G} \mid \mu_1 \longrightarrow t_2^{G} \mid \mu_2}{E[t_1^{G}] \mid \mu_1 \longmapsto E[t_2^{G}] \mid \mu_2} \qquad (\text{R}E\text{err})\ \frac{t_1^{G} \mid \mu \longrightarrow \textbf{error}}{E[t_1^{G}] \mid \mu \longmapsto \textbf{error}} \qquad (\text{R}F)\ \frac{et \longrightarrow_c et'}{F[et] \mid \mu \longmapsto F[et'] \mid \mu}$$

$$(\text{R}F\text{err})\ \frac{et \longrightarrow_c \textbf{error}}{F[et] \mid \mu \longmapsto \textbf{error}}$$

Fig. 5. $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$: Dynamic semantics.

The problem here is that $u \in \mathbb{T}[G_2]$, but o^{G_1} should map to some value of type G_1. We can extend the store as $\mu[o^{G_1} \mapsto \varepsilon u :: G_1]$, for some ε such that $G_2 \sim G_1$. Again, we combine evidences already present in the redex to construct new evidence. Notice that $\langle \text{Ref } G'_1\rangle \vdash \text{Ref } G_1 \sim \text{Ref } G_3$, then by an inversion lemma $\langle G'_1\rangle \vdash G_1 \sim G_3$. When considering consistency and not subtyping, evidence is also symmetric, then $\langle G'_1\rangle \vdash G_3 \sim G_1$. Also as $\langle G'_2\rangle \vdash G_2 \sim G_3$, by consistent transitivity $\langle G'_2\rangle \circ^{=} \langle G'_1\rangle \vdash G_2 \sim G_1$ (if defined). As $\langle G'_2\rangle \circ^{=} \langle G'_1\rangle = \langle G'_2 \sqcap G'_1\rangle$, then the final reduction rule is:

$$\langle G'_1\rangle o^{G_1} :=^{G_3} \langle G'_2\rangle u \mid \mu \longrightarrow \text{unit} \mid \mu[o^{G_1} \mapsto \langle G'_2 \sqcap G'_1\rangle u :: G_1]$$

if the meet is defined, and $\langle G'_1\rangle o^{G_1} :=^{G_3} \langle G'_2\rangle u \mid \mu \longrightarrow$ **error** otherwise.

4.5. Elaboration of $\lambda^{\varepsilon}_{\widetilde{REF}}$ terms

So far we have presented intrinsic terms without formally explaining how to derive them. Fig. 6 present the type-driven elaboration rules from $\lambda_{\widetilde{\text{REF}}}$ to $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$. Judgment $\Gamma; \Sigma \vdash t \rightsquigarrow_n t^{G} : G$[6] denotes the elaboration of term t^{G} from term t, where t is typed G under environments Γ and Σ. For simplicity, we write $t \rightsquigarrow_n t : G$ if $\cdot; \cdot \vdash t \rightsquigarrow_n t : G$. Basically each consistent type judgment is replaced by the initial evidence operator $\mathcal{I}_{=}$.

Rule $(TR\,::)$ recursively translates the subterm t, and the consistent subtyping judgment $G' \sim G$ from $(G\,::)$ is replaced with $\mathcal{I}_{=}(G', G)$, which computes evidence ε for consistent subtyping. This evidence is eventually placed next to the translated term $t^{G'}$. Most of the elaboration rules follow this same recipe. Rule $(TR\text{app})$, uses metafunctions $\widetilde{dom}$ and $\widetilde{cod}$ to avoid writing three different elaboration rules, e.g. when t_1 is typed $?$ then $\varepsilon_1 = \mathcal{I}_{=}(?, ? \to ?)$. The same is applied in rules $(TR\text{deref})$ and $(TR\text{asgn})$ where we use $\widetilde{tref}$ instead.

[6] We use the ε subindex to differentiate different translations presented in this chapter.

$$(T\mathrm{Rx})\ \frac{x : G \in \Gamma}{\Gamma ; \Sigma \vdash x \leadsto_n x^G : G} \qquad (T\mathrm{Rc})\ \frac{\theta(c) = B}{\Gamma ; \Sigma \vdash b \leadsto_n b : B}$$

$$(T\mathrm{Rapp})\ \frac{\begin{array}{cc}\Gamma ; \Sigma \vdash t_1 \leadsto_n t^{G_1} : G_1 & \Gamma ; \Sigma \vdash t_2 \leadsto_n t^{G_2} : G_2 \\ \varepsilon_1 = \mathcal{I}_=(G_1, \widetilde{dom}(G_1) \to \widetilde{cod}(G_1)) & \varepsilon_2 = \mathcal{I}_=(G_2, \widetilde{dom}(G_1))\end{array}}{\Gamma ; \Sigma \vdash t_1\ t_2 \leadsto_n \varepsilon_1 t^{G_1} \mathbin{@^{\widetilde{dom}(G_1) \to \widetilde{cod}(G_1)}} \varepsilon_2 t^{G_2} : \widetilde{cod}(G_1)}$$

$$(T\mathrm{Rop})\ \frac{\begin{array}{ccc}\Gamma ; \Sigma \vdash t_1 \leadsto_n t^{G_1} : G_1 & \Gamma ; \Sigma \vdash t_2 \leadsto_n t^{G_2} : G_2 & ty(\oplus) = B_1 \times B_2 \to B_3 \\ \varepsilon_1 = \mathcal{I}_=(G_1, B_1) & & \varepsilon_2 = \mathcal{I}_=(G_2, B_2)\end{array}}{\Gamma ; \Sigma \vdash t_1 \oplus t_2 \leadsto_n \varepsilon_1 t^{G_1} \oplus \varepsilon_2 t^{G_2} : B_3}$$

$$(T\mathrm{Rif})\ \frac{\begin{array}{c}\Gamma ; \Sigma \vdash t_1 \leadsto_n t^{G_1} : G_1 \quad \Gamma ; \Sigma \vdash t_2 \leadsto_n t^{G_2} : G_2 \quad \Gamma ; \Sigma \vdash t_3 \leadsto_n t^{G_3} : G_3 \\ \varepsilon_1 = \mathcal{I}_=(G_1, \mathsf{Bool}) \quad G = G_2 \sqcap G_3 \quad \varepsilon_2 = \mathcal{I}_=(G_2, G) \quad \varepsilon_3 = \mathcal{I}_=(G_3, G)\end{array}}{\Gamma ; \Sigma \vdash \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \leadsto_n \mathsf{if}\ \varepsilon_1 t^{G_1}\ \mathsf{then}\ \varepsilon_2 t^{G_2}\ \mathsf{else}\ \varepsilon_3 t^{G_3} : G}$$

$$(T\mathrm{R}\lambda)\ \frac{\Gamma, x : G_1 \vdash t \leadsto_n t^{G_2} : G_2}{\Gamma ; \Sigma \vdash \lambda x : G_1 . t \leadsto_n \lambda x^{G_1} . t^{G_2} : G_1 \to G_2} \qquad (T\mathrm{R::})\ \frac{\Gamma ; \Sigma \vdash t \leadsto_n t^{G'} : G' \quad \varepsilon = \mathcal{I}_=(G', G)}{\Gamma ; \Sigma \vdash (t :: G) \leadsto_n (\varepsilon t^{G'} :: G) : G}$$

$$(T\mathrm{Rref})\ \frac{\Gamma ; \Sigma \vdash t \leadsto_n t^G : G \quad \varepsilon = \mathcal{I}_=(G, G)}{\Gamma ; \Sigma \vdash \mathsf{ref}\ t \leadsto_n \mathsf{ref}^G\ \varepsilon t^G : \mathsf{Ref}\ G}$$

$$(T\mathrm{Rderef})\ \frac{\Gamma ; \Sigma \vdash t \leadsto_n t^{G'} : G' \quad G = \widetilde{tref}(G') \quad \varepsilon = \mathcal{I}_=(G', \mathsf{Ref}\ G)}{\Gamma ; \Sigma \vdash !t \leadsto_n !^G \varepsilon t^{G'} : G}$$

$$(T\mathrm{Rasgn})\ \frac{\begin{array}{cc}\Gamma ; \Sigma \vdash t_1 \leadsto_n t^{G_1} : G_1 & \Gamma ; \Sigma \vdash t_2 \leadsto_n t^{G_2} : G_2 \\ G_3 = \widetilde{tref}(G_1) \quad \varepsilon_1 = \mathcal{I}_=(G_1, \mathsf{Ref}\ G_3) & \varepsilon_2 = \mathcal{I}_=(G_2, G_3)\end{array}}{\Gamma ; \Sigma \vdash t_1 := t_2 \leadsto_n \varepsilon_1 t^{G_1} :=^{G_3} \varepsilon_2 t^{G_2} : \mathsf{Unit}} \qquad (T\mathrm{Rl})\ \frac{o : G \in \Sigma}{\Gamma ; \Sigma \vdash o \leadsto_n o^G : \mathsf{Ref}\ G}$$

Fig. 6. Elaboration of $\lambda^{\varepsilon}_{\widetilde{\mathrm{REF}}}$ from $\lambda_{\widetilde{\mathrm{REF}}}$.

Note that the elaboration rules only enrich derivations with evidence (by using the initial evidence operator), and such resulting derivations are represented as intrinsic terms. Then by construction, the elaboration rules trivially preserve typing:

Proposition 9 *(Elaboration preserves typing). If $\Gamma ; \Sigma \vdash t : G$ and $\Gamma ; \Sigma \vdash t \leadsto_n t^G : G$, then $t^G \in \mathbb{T}[G]$.*

4.6. Properties

In order to establish type safety we first have to define well-typedness of the store μ. Well-typedness of the store is usually defined with respect to a store environment, i.e. $\Sigma \vdash \mu$. Here, as we can see in Fig. 4, intrinsically-typed locations $o^G \in \mathbb{T}[\mathsf{Ref}\ G]$ obviate the need for store environment Σ: the store environment of a term t is simply the set of intrinsically-typed free locations of the term, $freeLocs(t)$. Therefore, contrary to standard reference type systems, well-typedness of the store is defined with respect to an intrinsic term:

Definition 8 *(μ is well typed).* A store μ is said to be *well typed* with respect to an intrinsic term t^G, written $t^G \vdash \mu$, if

1. $freeLocs(t^G) \subseteq dom(\mu)$, and
2. $\forall\ o^G \in dom(\mu),\ \mu(o^G) \in \mathbb{T}[G]$.

A store μ is well typed if all the free locations of a term are part of the domain of the store. Also for each of the intrinsic locations $o^G \in \mathbb{T}[G]$ that are part of the domain of the store, then all the underlying values $v \in \mathbb{T}[G]$.

Now we can establish type safety: closed terms do not get stuck, though they may terminate with cast errors. Also the store of a program is well typed.

Proposition 10 *(Type safety). Let t^G a closed intrinsic term. If $t^G \in \mathbb{T}[G]$ then one of the following is true:*

1. *t^G is a value v;*
2. *if $t^G \vdash \mu$ then $t^G \mid \mu \longmapsto t'^G \mid \mu'$ for some term $t'^G \in \mathbb{T}[G]$ and some μ' such that $t'^G \vdash \mu'$ and $dom(\mu) \subseteq dom(\mu')$;*
3. *$t^G \mid \mu \longmapsto$* **error**.

Also, the gradual type system is a conservative extension of the static type system; *i.e.* both systems coincide on fully-annotated terms (where every subterm has only static type annotations). We first present the conservative extension of the static semantics of the static language.

Proposition 11 *(Equivalence for fully-annotated terms (statics)). For any $t \in$ TERM, $. \vdash_s t : T$ if and only if $t : T$*

We now present the conservative extension of the dynamic semantics of the static language. The equivalence of the dynamic semantics for fully-annotated terms is more subtle. We cannot rely on a syntactic comparison of values because during reduction $\lambda^{\varepsilon}_{\widetilde{\mathrm{REF}}}$ inserts (possibly redundant) ascriptions. For instance, $(\lambda x : \mathsf{Int}.1)$ is syntactically different, but equivalent to $(\lambda x^{\mathsf{Int}}.\langle\mathsf{Int}\rangle 1 :: \mathsf{Int})$. To capture this relation, we formally connect both languages using logical relations between pairs of terms and stores.

We use notation $\langle t, \mu\rangle \approx \langle t^T, \mu'\rangle : T$ to denote that the pair of term t and store μ is related to the pair of term t^T and store μ' at type T. Two pairs of values and stores are related values at type T, if first, both stores are related. Two stores are related if for all locations that are common to both stores, the stored values are related. Second, if T is a constant B or a reference $\mathsf{Ref}\ T$, then both values must be equal. Third, if T is a function, then both functions applied to related arguments yield two related computations. Two configurations (*i.e.* term-store pairs) are related computations if both configurations reduce to related values and stores. The complete definition and proofs are presented in C.1.

Proposition 12 *(Equivalence for fully-annotated terms (dynamics)). For any $t \in$ TERM, $. \vdash_s t : T$, $t \rightsquigarrow_n t^T : T$, then $t \mid \cdot \longmapsto^*_s v \mid \mu \iff t^T \mid \cdot \longmapsto^* v' \mid \mu'$, for some μ, μ' such that $\langle v, \mu\rangle \approx \langle v', \mu'\rangle : T$.*

Precision on terms, noted $t_1 \sqsubseteq t_2$, is the natural lifting of type precision to terms. The gradual type system satisfies the static gradual guarantee of Siek et al. [38], *i.e.* losing precision preserves typeability: if a program is well-typed, then a less precise version of it also type checks, at a less precise type.

Proposition 13 *(Static gradual guarantee). If $t_1 : G_1$ and $t_1 \sqsubseteq t_2$, then $t_2 : G_2$, for some G_2 such that $G_1 \sqsubseteq G_2$.*

We also prove that $\lambda_{\widetilde{\mathrm{REF}}}$ satisfies the dynamic component of the gradual guarantee: "any program that runs without error would continue to do so if it were given less precise types". For this we must also extend the notion of precision over stores: intuitively a store is more precise than another store if its locations and values are more precise than the locations and values of the other.

Proposition 14 *(Dynamic gradual guarantee). Suppose $t_1^{G_1} \sqsubseteq t_1^{G_2}$ and $\mu_1 \sqsubseteq \mu_2$. Then if $t_1^{G_1} \mid \mu_1 \longmapsto t_2^{G_1} \mid \mu_1'$ then $t_1^{G_2} \mid \mu_2 \longmapsto t_2^{G_2} \mid \mu_2'$ where $t_2^{G_1} \sqsubseteq t_2^{G_2}$ and $\mu_1' \sqsubseteq \mu_2'$.*

4.7. $\lambda_{\widetilde{\mathrm{REF}}}$ in action

$\lambda_{\widetilde{\mathrm{REF}}}$ is semantically equivalent to HCC. The resulting language $\lambda_{\widetilde{\mathrm{REF}}}$ behaves exactly as HCC. Recall the examples from §2.3; $\lambda_{\widetilde{\mathrm{REF}}}$, like HCC, rejects examples 2 and 4, and accepts examples 3 and 5.

For instance, consider example 2. The corresponding $\lambda_{\widetilde{\mathrm{REF}}}$ term is $!((\mathsf{ref}\ 4 :: ?) :: \mathsf{Ref\ Bool})$. Its elaboration reduces as follows:

$$\begin{aligned}
&!^{\mathsf{Bool}}\langle\mathsf{Ref\ Bool}\rangle(\langle\mathsf{Ref\ Bool}\rangle(\mathsf{ref}^{?}\ \langle\mathsf{Int}\rangle 4 :: ?) :: \mathsf{Ref\ Bool}) \mid \cdot\\
\longmapsto\ &!^{\mathsf{Bool}}\langle\mathsf{Ref\ Bool}\rangle(\langle\mathsf{Ref\ Bool}\rangle o^{?} :: \mathsf{Ref\ Bool}) \mid [o^{?} \mapsto \langle\mathsf{Int}\rangle 4 :: ?]\\
\longmapsto\ &!^{\mathsf{Bool}}\langle\mathsf{Ref\ Bool}\rangle o^{?} \mid [o^{?} \mapsto \langle\mathsf{Int}\rangle 4 :: ?]\\
\longmapsto\ &\langle\mathsf{Bool}\rangle(\langle\mathsf{Int}\rangle 4 :: ?) :: \mathsf{Bool} \mid [o^{?} \mapsto \langle\mathsf{Int}\rangle 4 :: ?]\\
\longmapsto\ &\mathbf{error}
\end{aligned}$$

because $\langle\mathsf{Int}\rangle \circ \langle\mathsf{Bool}\rangle$ is not defined. Of course, this is just an example reduction; formally establishing the relation between both languages is the subject of §5.

$\lambda_{\widetilde{\mathrm{REF}}}$ is not space efficient. Even though semantically equivalent, contrary to HCC, $\lambda_{\widetilde{\mathrm{REF}}}$ is not *space efficient*. We can write programs in $\lambda_{\widetilde{\mathrm{REF}}}$ that accumulate an unbounded number of evidences during reduction.

To illustrate, consider $\lambda_{\widetilde{\mathrm{REF}}}$ term $\Omega = (\lambda x : ?.x\,x)(\lambda x : ?.x\,x)$. Its elaboration to $\lambda^{\varepsilon}_{\widetilde{\mathrm{REF}}}$ is

$$\Omega^{?} = \langle ? \to ?\rangle(\lambda x^{?}.\langle ? \to ?\rangle x\,\langle ?\rangle x) \mathbin{@^{? \to ?}} \langle ? \to ?\rangle(\lambda x^{?}.\langle ? \to ?\rangle x\,\langle ?\rangle x)$$

After multiple steps of reduction, the resulting term accumulates ascriptions as follows:

$$\langle ?\rangle(\langle ?\rangle(\langle ?\rangle \ldots (\langle ?\rangle \Omega^{?} :: ?) \ldots :: ?) :: ?) :: ?$$

This example illustrates how the order of combination of evidences impacts space efficiency and destroys tail recursion. Note that this problem applies to any language derived with AGT.

In contrast, with HCC, the same program reduces as follows (we omit stores for simplicity):

$$\mathbf{\Omega}^{?} = (\lambda x : ?.\mathbf{c_1 x x})\, \mathbf{c_2}(\lambda x : ?.\mathbf{c_1 x x}) \longmapsto \mathbf{c_1}(\mathbf{c_2}(\lambda x : ?.\mathbf{c_1 x x}))\, \mathbf{c_2}(\lambda x : ?.\mathbf{c_1 x x}) \longmapsto \mathbf{\Omega}^{?} \longmapsto \ldots$$

where $\mathbf{c_1}$ is a coercion from $?$ to $? \to ?$, and $\mathbf{c_2}$ from $? \to ?$ to $?$.

Even though $\lambda_{\widetilde{\text{REF}}}$ and HCC are different regarding space efficiency, they are semantically equivalent: given a term and its compilations to $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ and HCC^{+} (an adapted version of HCC), either both terms reduce to values, both terms diverge, or both terms reduce to an error. In the following, we formalize the relation between $\lambda_{\widetilde{\text{REF}}}$ and HCC (§5), along with the changes needed to recover space efficiency in $\lambda_{\widetilde{\text{REF}}}$ (§5.3).

5. Comparing $\lambda_{\widetilde{\text{REF}}}$ and HCC

In this section we compare $\lambda_{\widetilde{\text{REF}}}$ and HCC, the space-efficient coercion calculus of Herman et al. [24]. We start by presenting the static and dynamic semantics of HCC^{+}, an adapted version of HCC extended with conditionals and binary operations. Then we formalize the relation between both semantics as follows: given a $\lambda_{\widetilde{\text{REF}}}$ term and its corresponding elaboration to $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ and translation to HCC^{+}, we prove that the resulting terms are bisimilar, and that consequently they either both terminate, both fail, or both diverge. Despite this tight relation, the dynamic semantics of $\lambda_{\widetilde{\text{REF}}}$ are not space-efficient: ascriptions can be repeatedly accumulated during reduction, contrary to HCC. We finalize this section by adjusting the dynamic semantics of $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ to recover space efficiency.

5.1. The coercion calculus

In this section we present HCC^{+}, an adaptation of HCC extended with conditionals and binary operations. This language is designed as a cast calculus for $\lambda_{\widetilde{\text{REF}}}$. The following presentation of this language is closely related to the coercion calculus presented by Siek et al. [37].

Usually, the operational semantics of gradual languages generate proxies when reducing function applications which involve casts. This approach may result in an unbounded growth in the number of proxies, which impacts space efficiency and destroys tail recursion [24]. HCC was designed to represent and compress sequences of casts, by using coercions instead of casts (and function proxies). HCC recovers space efficiency by combining and normalizing adjacent coercions to limit their space consumption to a constant factor.

Static semantics. Fig. 7 presents the static semantics of HCC^{+}. The syntax includes gradual types G, ground types R, coercions $\mathbf{c}$, and terms $\mathbf{t}$. Ground types R are the only types allowed to be coerced directly from/to the unknown type $?$. A ground type can be either a function $? \to ?$, a reference $\mathsf{Ref}\ ?$, or a base type B. Terms $\mathbf{t}$ can also be coerced terms $\mathbf{ct}$. The judgment $\mathbf{c} \vdash G_1 \Rightarrow G_2$ represents that coercion $\mathbf{c}$ is used to coerce values of type G_1 to type G_2. The identity coercion $\iota_{\mathbf{G}}$ represents a coercion from a type to itself. The failure coercion Fail represents an invalid coercion. The *tagging* coercion $\mathbf{R!}$ represents a coercion from a ground type R to $?$. The *check-and-untag* coercion $\mathbf{R?}$ represents a coercion from $?$ to a ground type R. The function coercion $\mathbf{c_1} \to \mathbf{c_2}$ represents a coercion where $\mathbf{c_1}$ coerces the function argument, and $\mathbf{c_2}$ coerces the result. The reference coercion $\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2}$ represents a coercion where $\mathbf{c_1}$ coerces values written in the heap, and $\mathbf{c_2}$ coerces values read from the heap. Finally, a coercion composition $\mathbf{c_1}; \mathbf{c_2}$ represents the coercion $\mathbf{c_1}$ followed by coercion $\mathbf{c_2}$. We consider coercions equal up to associativity of composition. The type rules are standard for a cast calculus. Each type rule of Fig. 3 is simplified by replacing uses of consistency with equality. We replace the ascription rule (G::) with rule ($\mathbf{H}C$), used to typecheck coerced terms: a coerced term $\mathbf{ct}$ has type G if the subterm $\mathbf{t}$ has type G', and $\mathbf{c}$ is a coercion from G' to G.

Dynamic semantics. Fig. 8 presents the dynamic semantics of HCC^{+}. A value $\mathbf{v}$ can be a raw value $\mathbf{u}$, or a coerced value $\mathbf{c\,u}$, where coercion $\mathbf{c}$ is in *normal form*. We say a coercion is in normal form if it is irreducible, denoted $\mathsf{nm}\ \mathbf{c}$; the predicate is defined in Fig. 9. To reduce programs we use three different evaluation contexts: $\mathbf{H}$ to reduce coercions, and $\mathbf{F}$ and $\mathbf{E}$ to reduce terms. Coercions are combined using the *coerced term reduction* rule $\longrightarrow_c$. Coercions are maintained in normal form throughout evaluation using big step semantics of the *coercion reduction* rule $\longmapsto$, and the normal form predicate $\mathsf{nm}\ \mathbf{c}$. The coercion reduction rule combines coercions using the *notion of coercion reduction* rule $\longrightarrow$. A failure coercion is produced when a tagging and a check-and-untag coercions are combined, and the ground types involved are different. When the types are the same, then an identity coercion is produced. The combination of an identity coercion with another coercion $\mathbf{c}$ produces the same coercion $\mathbf{c}$. On the contrary, the combination of a failure coercion with another coercion $\mathbf{c}$ propagates the failure coercion. Reductions of both combination of function coercions and combination of reference coercions are defined inductively. Notice the contravariant combination order for the argument of functions, and in the coercions for write values in the heap respectively. Note that in the reduction of coerced terms, a failure coercion does not trigger a runtime error immediately (i.e. $\mathsf{Fail}\,\mathbf{t} \longrightarrow_c \mathbf{error}$), but after the subterm is reduced to a raw value (i.e. $\mathsf{Fail}\,\mathbf{u} \longrightarrow_c \mathbf{error}$). The reason for this is that HCC aims to regain space efficiency without changing the behavior of standard cast calculus/coercion semantics, which combines casts when the subterm is a value. The rest of the dynamic semantics is standard to cast calculi. The reduction of coerced dereferences coerces the value on the heap with the second component of the coercion, and dually the reduction of coerced assignments coerces the updated value using the first component of the coercion.

$$R \in \text{GroundType}, \quad \mathbf{c} \in \text{Coercion}, \quad \mathbf{t} \in \text{CTerm},$$

$$\begin{array}{lll}
G & ::= & ? \mid B \mid G \to G \mid \mathsf{Ref}\ G & \text{(Gradual types)} \\
R & ::= & ? \to ? \mid \mathsf{Ref}\ ? \mid B & \text{(Ground types)} \\
\mathbf{c} & ::= & \iota_{\mathbf{G}} \mid \mathsf{Fail} \mid \mathbf{R!} \mid \mathbf{R?} \mid \mathbf{c} \to \mathbf{c} \mid \mathsf{Ref}\ \mathbf{c}\ \mathbf{c} \mid \mathbf{c}; \mathbf{c} & \text{(coercions)} \\
\mathbf{t} & ::= & b \mid (\lambda x : G.\mathbf{t}) \mid o \mid x \mid \mathbf{t}\ \mathbf{t} \mid \mathbf{t} \oplus \mathbf{t} \mid \mathsf{if}\ \mathbf{t}\ \mathsf{then}\ \mathbf{t}\ \mathsf{else}\ \mathbf{t} \mid \mathbf{c}\ \mathbf{t} \mid \mathsf{ref}\ \mathbf{t} \mid !\mathbf{t} \mid \mathbf{t} {:=} \mathbf{t} & \text{(terms)}
\end{array}$$

$\boxed{\mathbf{c} \vdash G_1 \Rightarrow G_2}$ **Coercion typing**

$$\frac{}{\iota_{\mathbf{G}} \vdash G \Rightarrow G} \qquad \frac{}{\mathsf{Fail} \vdash G_1 \Rightarrow G_2} \qquad \frac{}{\mathbf{R?} \vdash ? \Rightarrow R} \qquad \frac{}{\mathbf{R!} \vdash R \Rightarrow ?}$$

$$\frac{\mathbf{c_1} \vdash G_{21} \Rightarrow G_{11} \qquad \mathbf{c_2} \vdash G_{12} \Rightarrow G_{22}}{\mathbf{c_1} \to \mathbf{c_2} \vdash G_{11} \to G_{12} \Rightarrow G_{21} \to G_{22}} \qquad \frac{\mathbf{c_1} \vdash G_2 \Rightarrow G_1 \qquad \mathbf{c_2} \vdash G_1 \Rightarrow G_2}{\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2} \vdash \mathsf{Ref}\ G_1 \Rightarrow \mathsf{Ref}\ G_2}$$

$$\frac{\mathbf{c_1} \vdash G_1 \Rightarrow G_2 \qquad \mathbf{c_2} \vdash G_2 \Rightarrow G_3}{\mathbf{c_1}; \mathbf{c_2} \vdash G_1 \Rightarrow G_3}$$

$\boxed{\Gamma; \Sigma \vdash_H \mathbf{t} : G}$ **Term typing**

$$(\mathbf{H}x)\frac{x : G \in \Gamma}{\Gamma; \Sigma \vdash_H x : G} \qquad (\mathbf{H}b)\frac{\theta(b) = B}{\Gamma; \Sigma \vdash_H b : B} \qquad (\mathbf{H}\text{app})\frac{\Gamma; \Sigma \vdash_H \mathbf{t_1} : G_1 \to G_2 \qquad \Gamma; \Sigma \vdash_H \mathbf{t_2} : G_1}{\Gamma; \Sigma \vdash_H \mathbf{t_1}\ \mathbf{t_2} : G_2}$$

$$(\mathbf{H}\text{op})\frac{\begin{array}{c}\Gamma; \Sigma \vdash_H t_1 : B_1 \qquad \Gamma; \Sigma \vdash_H t_2 : B_2 \\ ty(\oplus) = B_1 \times B_2 \to B_3\end{array}}{\Gamma; \Sigma \vdash_H \mathbf{t_1} \oplus \mathbf{t_2} : B_3} \qquad (\mathbf{H}\text{if})\frac{\begin{array}{c}\Gamma; \Sigma \vdash_H \mathbf{t_1} : \mathsf{Bool} \qquad \Gamma; \Sigma \vdash_H \mathbf{t_2} : G_2 \\ \Gamma; \Sigma \vdash_H \mathbf{t_3} : G_2\end{array}}{\Gamma; \Sigma \vdash_H \mathsf{if}\ \mathbf{t_1}\ \mathsf{then}\ \mathbf{t_2}\ \mathsf{else}\ \mathbf{t_3} : G_2}$$

$$(\mathbf{H}\lambda)\frac{\Gamma, x : G_1 \vdash_H \mathbf{t} : G_2}{\Gamma; \Sigma \vdash_H (\lambda x : G_1.\mathbf{t}) : G_1 \to G_2} \qquad (\mathbf{H}C)\frac{\Gamma; \Sigma \vdash_H \mathbf{t} : G' \qquad \mathbf{c} \vdash G' \Rightarrow G}{\Gamma; \Sigma \vdash_H \mathbf{c}\ \mathbf{t} : G}$$

$$(\mathbf{H}\text{ref})\frac{\Gamma; \Sigma \vdash_H \mathbf{t} : G}{\Gamma; \Sigma \vdash_H \mathsf{ref}\ \mathbf{t} : \mathsf{Ref}\ G} \qquad (\mathbf{H}\text{deref})\frac{\Gamma; \Sigma \vdash_H \mathbf{t} : \mathsf{Ref}\ G}{\Gamma; \Sigma \vdash_H !\mathbf{t} : G}$$

$$(\mathbf{H}\text{asgn})\frac{\Gamma; \Sigma \vdash_H \mathbf{t_1} : \mathsf{Ref}\ G \qquad \Gamma; \Sigma \vdash_H \mathbf{t_2} : G}{\Gamma; \Sigma \vdash_H \mathbf{t_1} {:=} \mathbf{t_2} : \mathsf{Unit}} \qquad (\mathbf{H}\text{l})\frac{o : G \in \Sigma}{\Gamma; \Sigma \vdash_H o : \mathsf{Ref}\ G}$$

Fig. 7. HCC$^+$: Static semantics.

Translation semantics. Fig. 10 presents the translation rules from $\lambda_{\widetilde{\text{REF}}}$ to HCC$^+$. The translation is a type-driven coercion insertion. The key idea is to insert coercions where consistency is used in the typing derivation. The translation judgment has the form $\Gamma; \Sigma \vdash t \rightsquigarrow_c \mathbf{t'} : G$ which represent translation from $\lambda_{\widetilde{\text{REF}}}$ term t of type G, to HCC$^+$ term $\mathbf{t'}$, under environments Γ and Σ. We write $t \rightsquigarrow_n \mathbf{t} : G$ if $\cdot; \cdot \vdash t \rightsquigarrow_n \mathbf{t} : G$. Note that we assume that variables x and x refer the same variable. Similarly, constants b and b, and locations o and o refer to the same location. We paint them red only to disambiguate terms of different languages. Coercions are introduced using the $\langle G_1 \Rightarrow G_2 \rangle \mathbf{t}$ metafunction, which represents the insertion of a coercion from G_1 to G_2. This metafunction avoids the insertion of redundant coercions by checking if G_1 is syntactically equal to G_2. If both types are the same, then the coercion is not introduced. Otherwise we use the coercion function $\langle\!\langle G_1 \Rightarrow G_2 \rangle\!\rangle$ to elaborate the coercion from G_1 to G_2. The inductive definition of the coercion inserting function is presented in Fig. 11, and follows closely the rules for coercion typing. For instance, as $\mathbf{R?} \vdash ? \Rightarrow R$, then $\langle\!\langle ? \Rightarrow R \rangle\!\rangle = \mathbf{R?}$. There are some subtleties worth mentioning, such as the definition of $\langle\!\langle ? \Rightarrow \mathsf{Ref}\ G \rangle\!\rangle$. This should result in a coercion from $?$ to $\mathsf{Ref}\ G$, but there is no direct coercion from unknown to any given type $\mathsf{Ref}\ G$. Consequently $\langle\!\langle ? \Rightarrow \mathsf{Ref}\ G \rangle\!\rangle$ is defined as the composition of a coercion from $?$ to $\mathsf{Ref}\ ?$: $(\mathsf{Ref}\ ?)?$ (to test if the value is actually a reference), with a coercion from $\mathsf{Ref}\ ?$ to $\mathsf{Ref}\ G$: $\langle\!\langle \mathsf{Ref}\ ? \Rightarrow \mathsf{Ref}\ G \rangle\!\rangle$, which follows the inductive definition. Analogously, $\langle\!\langle \mathsf{Ref}\ G \Rightarrow ? \rangle\!\rangle$ is defined as the composition of a coercion from $\mathsf{Ref}\ G$ to $\mathsf{Ref}\ ?$, with a coercion from $\mathsf{Ref}\ ?$ to $?$. Notice that by construction, we do not need definitions for $\langle\!\langle G \Rightarrow G \rangle\!\rangle$ and $\langle\!\langle ? \Rightarrow ? \rangle\!\rangle$ as these cases are avoided thanks to the $\langle . \Rightarrow . \rangle$ metafunction.

5.2. Relating $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ and HCC$^+$

We now establish the equivalence of the $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ and HCC$^+$ semantics for elaborated and translated $\lambda_{\widetilde{\text{REF}}}$ terms respectively, by using a bisimulation relation.

Fig. 12 presents function $(\!|.|\!)$, which relates evidence augmented consistent judgments with coercions during the definition of the bisimulation relation. A naive relation between evidence and coercion is ambiguous unless one indicates the gradual types involved in the judgment. For instance, evidence $\langle \mathsf{Int} \rangle$ corresponds to both coercion $\mathsf{Int?}$ or $\mathsf{Int!}$, unless we expose the judgment associated with the evidence, so $\langle \mathsf{Int} \rangle \vdash \mathsf{Int} \sim ?$ correspond exactly to the coercion from Int to $?$, $\mathsf{Int!}$. The definition follows the definition of the coercion insertion function presented in Fig. 10 (e.g. $\langle\!\langle ? \Rightarrow R \rangle\!\rangle = \mathbf{R?}$, so

$$
\begin{array}{rcll}
\mathbf{u} & ::= & b \mid (\lambda x : G_1.\mathbf{t}) \mid o & \text{(raw values)}\\
\mathbf{v} & ::= & \mathbf{u} \mid \mathbf{c}\,\mathbf{u} \text{ where nm } \mathbf{c} & \text{(values)}\\
\mathbf{H} & ::= & \square \mid \square; \mathbf{c} \mid \mathbf{c}; \square \mid \square \to \mathbf{c} \mid \mathbf{c} \to \square \mid \mathsf{Ref}\ \square\ \mathbf{c} \mid \mathsf{Ref}\ \mathbf{c}\ \square & \text{(Coercion contexts)}\\
\mathbf{F} & ::= & \square \mid \mathbf{E} + \mathbf{t} \mid \mathbf{v} + \mathbf{E} \mid \mathbf{E}\ \mathbf{t} \mid \mathbf{v}\ \mathbf{E} \mid \mathbf{E} :: G \mid & \\
 & & \mathsf{if}\ \mathbf{E}\ \mathsf{then}\ \mathbf{t}\ \mathsf{else}\ \mathbf{t} \mid \mathsf{ref}\ \mathbf{E} \mid !\mathbf{E} \mid \mathbf{E}{:=}\mathbf{t} \mid \mathbf{v}{:=}\mathbf{E} & \text{(Cast-free contexts)}\\
\mathbf{E} & ::= & \mathbf{F} \mid \mathbf{c}\ \mathbf{F} & \text{(Evaluation contexts)}
\end{array}
$$

$\boxed{\longrightarrow : \textsc{Coercion} \times \textsc{Coercion}}$

Notion of coercion reduction

$$
\begin{array}{rcll}
\mathbf{R}!; \mathbf{R}? & \longrightarrow & \iota_{\mathbf{R}} & \\
\mathbf{R_1}!; \mathbf{R_2}? & \longrightarrow & \mathsf{Fail} & \text{if } \mathbf{R_1} \neq \mathbf{R_2}\\
(\mathbf{c_{11}} \to \mathbf{c_{12}}); (\mathbf{c_{21}} \to \mathbf{c_{22}}) & \longrightarrow & (\mathbf{c_{21}}; \mathbf{c_{11}}) \to (\mathbf{c_{12}}; \mathbf{c_{22}}) & \\
(\mathsf{Ref}\ \mathbf{c_{11}}\ \mathbf{c_{12}}); (\mathsf{Ref}\ \mathbf{c_{21}}\ \mathbf{c_{22}}) & \longrightarrow & \mathsf{Ref}\ (\mathbf{c_{21}}; \mathbf{c_{11}})\ (\mathbf{c_{12}}; \mathbf{c_{22}}) & \\
\iota_G; \mathbf{c} & \longrightarrow & \mathbf{c} & \\
\mathbf{c}; \iota_G & \longrightarrow & \mathbf{c} & \\
\mathsf{Fail}; \mathbf{c} & \longrightarrow & \mathsf{Fail} & \\
\mathbf{c}; \mathsf{Fail} & \longrightarrow & \mathsf{Fail} &
\end{array}
$$

$\boxed{\longmapsto : \textsc{Coercion} \times \textsc{Coercion}}$

Coercion reduction

$$
\frac{\mathbf{c_1} \longrightarrow \mathbf{c_2}}{\mathbf{H}[\mathbf{c_1}] \longmapsto \mathbf{H}[\mathbf{c_2}]}
$$

$\boxed{\longrightarrow_c : \textsc{CTerm} \times (\textsc{CTerm} \cup \{\mathbf{error}\})}$

Coerced term reduction

$$
\begin{array}{rcl}
\iota_G\,\mathbf{u} \mid \mu & \longrightarrow_c & \mathbf{u} \mid \mu \\
\mathbf{c_1}\ (\mathbf{c_2}\ \mathbf{t}) \mid \mu & \longrightarrow_c & \mathbf{c}\ \mathbf{t} \mid \mu \\
 & & \quad \text{if } \mathbf{c_2}; \mathbf{c_1} \longmapsto^* \mathbf{c} \wedge \text{nm } \mathbf{c}\\
\mathsf{Fail}\,\mathbf{u} & \longrightarrow_c & \mathbf{error}
\end{array}
$$

$\textsc{Config} = \textsc{CTerm} \times \textsc{Store}$

$\boxed{\longrightarrow : \textsc{Config} \times \textsc{Config}}$

Notion of term reduction

$$
\begin{array}{rcll}
(\lambda x : G.\mathbf{t})\ \mathbf{v} \mid \mu & \longrightarrow & \mathbf{t}[\mathbf{v}/x] \mid \mu & \\
((\mathbf{c_1} \to \mathbf{c_2})\ \mathbf{u})\mathbf{v} \mid \mu & \longrightarrow & \mathbf{c_1}\ (\mathbf{u}\ (\mathbf{c_2}\ \mathbf{v})) \mid \mu & \\
\mathsf{ref}\ \mathbf{v} \mid \mu & \longrightarrow & o \mid \mu[o \mapsto \mathbf{v}] & o \text{ fresh}\\
!o \mid \mu & \longrightarrow & \mu(o) \mid \mu & \\
o{:=}\mathbf{v} \mid \mu & \longrightarrow & \mathsf{unit} \mid \mu[o \mapsto \mathbf{v}] & \\
!((\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2})\ o) \mid \mu & \longrightarrow & \mathbf{c_2}\ !o \mid \mu & \\
(\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2})\ o{:=}\mathbf{v} \mid \mu & \longrightarrow & (o{:=}\mathbf{c_1}\ \mathbf{v}) \mid \mu & \\
\mathsf{if}\ \mathbf{b}\ \mathsf{then}\ \mathbf{t_2}\ \mathsf{else}\ \mathbf{t_3} \mid \mu & \longrightarrow & \begin{cases} \mathbf{t_2} \mid \mu \text{ if } b = \mathsf{true} \\ \mathbf{t_3} \mid \mu \text{ if } b = \mathsf{false} \end{cases} & \\
\mathbf{u_1} \oplus \mathbf{u_2} \mid \mu & \longrightarrow & \mathbf{u_1}\ [\![\oplus]\!]\ \mathbf{u_2} &
\end{array}
$$

$\boxed{\longmapsto : \textsc{Config} \times (\textsc{Config} \cup \{\mathbf{error}\})}$

Term reduction

$$
\frac{\mathbf{t} \mid \mu \longrightarrow \mathbf{t}' \mid \mu'}{\mathbf{E}[\mathbf{t}] \mid \mu \longmapsto \mathbf{E}[\mathbf{t}'] \mid \mu'}
\qquad
\frac{\mathbf{t} \mid \mu \longmapsto \mathbf{error}}{\mathbf{E}[\mathbf{t}] \mid \mu \longmapsto \mathbf{error}}
$$

$$
\frac{\mathbf{t} \longrightarrow_c \mathbf{t}'}{\mathbf{F}[\mathbf{t}] \mid \mu \longmapsto \mathbf{F}[\mathbf{t}'] \mid \mu}
\qquad
\frac{\mathbf{t} \longrightarrow_c \mathbf{error}}{\mathbf{F}[\mathbf{t}] \mid \mu \longmapsto \mathbf{error}}
$$

Fig. 8. HCC^{+}: Dynamic semantics.

$\boxed{\text{nm } \mathbf{c}}$

$$
\text{nm } \iota_G \qquad \text{nm } \mathsf{Fail} \qquad \text{nm } \mathbf{R}? \qquad \text{nm } \mathbf{R}! \qquad \text{nm } \mathbf{R}?; \mathbf{R}! \qquad \frac{\text{nm } \mathbf{c_1} \quad \text{nm } \mathbf{c_2}}{\mathbf{c_1} \to \mathbf{c_2}} \qquad \frac{\text{nm } \mathbf{c_1} \quad \text{nm } \mathbf{c_2}}{\text{nm } (? \to ?)?; \mathbf{c_1} \to \mathbf{c_2}}
$$

$$
\frac{\text{nm } \mathbf{c_1} \quad \text{nm } \mathbf{c_2}}{\text{nm } \mathbf{c_1} \to \mathbf{c_2}; (? \to ?)!} \qquad \frac{\text{nm } \mathbf{c_1} \quad \text{nm } \mathbf{c_2}}{\text{nm } (? \to ?)?; \mathbf{c_1} \to \mathbf{c_2}; (? \to ?)!} \qquad \frac{\text{nm } \mathbf{c_1} \quad \text{nm } \mathbf{c_2}}{\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2}}
$$

$$
\frac{\text{nm } \mathbf{c_1} \quad \text{nm } \mathbf{c_2}}{\text{nm } (\mathsf{Ref}\ ?)?; \mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2}} \qquad \frac{\text{nm } \mathbf{c_1} \quad \text{nm } \mathbf{c_2}}{\text{nm } \mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2}; (\mathsf{Ref}\ ?)!} \qquad \frac{\text{nm } \mathbf{c_1} \quad \text{nm } \mathbf{c_2}}{\text{nm } (\mathsf{Ref}\ ?)?; \mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2}; (\mathsf{Ref}\ ?)!}
$$

Fig. 9. HCC^{+}: Coercion normal forms.

$(\!|\langle R\rangle \vdash ? \sim R|\!) = \mathbf{R}?$), save for a few extra cases described next. Reflexive judgments on ground types and the unknown type, where evidence corresponds to the initial evidence, are mapped to identity coercions for ground types and the unknown type respectively. The definition also takes into consideration judgments where both types are unknown, and the evidence

$\boxed{\Gamma; \Sigma \vdash t \leadsto_c \mathbf{t}' : G}$ **Translation rules**

$$(HR\text{x})\,\frac{x : G \in \Gamma}{\Gamma; \Sigma \vdash x \leadsto_c x : G} \qquad (HR\text{c})\,\frac{\theta(b) = B}{\Gamma; \Sigma \vdash b \leadsto_c b : B}$$

$$(HR\text{app})\,\frac{\Gamma; \Sigma \vdash t_1 \leadsto_c \mathbf{t_1'} : G_1 \qquad \Gamma; \Sigma \vdash t_2 \leadsto_c \mathbf{t_2'} : G_2}{\Gamma; \Sigma \vdash t_1\ t_2 \leadsto_c \langle G_1 \Rightarrow \widetilde{dom}(G_1) \to \widetilde{cod}(G_2)\rangle \mathbf{t_1'}\ \langle G_2 \Rightarrow \widetilde{dom}(G_1)\rangle \mathbf{t_2'} : \widetilde{cod}(G_1)}$$

$$(HR\text{op})\,\frac{\Gamma; \Sigma \vdash t_1 \leadsto_c \mathbf{t_1'} : G_1 \qquad \Gamma; \Sigma \vdash t_2 \leadsto_c \mathbf{t_2'} : G_2 \qquad ty(\oplus) = B_1 \times B_2 \to B_3}{\Gamma; \Sigma \vdash t_1 \oplus t_2 \leadsto_c \langle G_1 \Rightarrow B_1\rangle \mathbf{t_1'} \oplus \langle G_2 \Rightarrow B_2\rangle \mathbf{t_2'} : B_3}$$

$$(HR\text{if})\,\frac{\Gamma; \Sigma \vdash t_1 \leadsto_c \mathbf{t_1'} : G_1 \qquad \Gamma; \Sigma \vdash t_2 \leadsto_c \mathbf{t_2'} : G_2 \qquad \Gamma; \Sigma \vdash t_3 \leadsto_c \mathbf{t_3'} : G_3 \qquad G = G_2 \sqcap G_3}{\Gamma; \Sigma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \leadsto_c \text{if } \langle G_1 \Rightarrow \text{Bool}\rangle \mathbf{t_1'} \text{ then } \langle G_2 \Rightarrow G\rangle \mathbf{t_2'} \text{ else } \langle G_3 \Rightarrow G\rangle \mathbf{t_3'} : G}$$

$$(HR\lambda)\,\frac{\Gamma, x : G_1 \vdash t \leadsto_c \mathbf{t_2'} : G_2}{\Gamma; \Sigma \vdash (\lambda x : G_1.t) \leadsto_c (\lambda x : G_1.\mathbf{t_2'}) : G_1 \to G_2} \qquad (HR\text{::})\,\frac{\Gamma; \Sigma \vdash t \leadsto_c \mathbf{t}' : G'}{\Gamma; \Sigma \vdash (t :: G) \leadsto_c \langle G' \Rightarrow G\rangle \mathbf{t}' : G}$$

$$(HR\text{ref})\,\frac{\Gamma; \Sigma \vdash t \leadsto_c \mathbf{t}' : G'}{\Gamma; \Sigma \vdash \text{ref } t \leadsto_c \text{ref } \langle G' \Rightarrow G\rangle \mathbf{t}' : \text{Ref } G} \qquad (HR\text{deref})\,\frac{\Gamma; \Sigma \vdash t \leadsto_c \mathbf{t}' : G' \qquad G = \widetilde{tref}(G')}{\Gamma; \Sigma \vdash !t \leadsto_c !\langle G' \Rightarrow \text{Ref } G\rangle \mathbf{t}' : G}$$

$$(HR\text{asgn})\,\frac{\Gamma; \Sigma \vdash t_1 \leadsto_c \mathbf{t_1'} : G_1 \qquad \Gamma; \Sigma \vdash t_2 \leadsto_c \mathbf{t_2'} : G_2 \qquad G_3 = \widetilde{tref}(G_1)}{\Gamma; \Sigma \vdash t_1 := t_2 \leadsto_c \langle G_1 \Rightarrow \text{Ref } G_3\rangle \mathbf{t_1'} := \langle G_2 \Rightarrow G_3\rangle \mathbf{t_2'} : \text{Unit}} \qquad (HR\text{l})\,\frac{o : G \in \Sigma}{\Gamma; \Sigma \vdash o \leadsto_c o : \text{Ref } G}$$

$$\text{where } \langle G_1 \Rightarrow G_2\rangle \mathbf{t} = \begin{cases} \mathbf{t} & \text{if } G_1 = G_2 \\ \langle\!\langle G_1 \Rightarrow G_2\rangle\!\rangle \mathbf{t} & \text{otherwise} \end{cases}$$

Fig. 10. $\lambda_{\widetilde{\text{REF}}}$ to HCC$^+$ translation rules.

$$\langle\!\langle ? \Rightarrow R\rangle\!\rangle = \mathbf{R?} \qquad \langle\!\langle R \Rightarrow ?\rangle\!\rangle = \mathbf{R!} \qquad \langle\!\langle ? \Rightarrow G_1 \to G_2\rangle\!\rangle = (? \to ?)?; \langle\!\langle ? \to ? \Rightarrow G_1 \to G_2\rangle\!\rangle$$

$$\langle\!\langle G_1 \to G_2 \Rightarrow ?\rangle\!\rangle = \langle\!\langle G_1 \to G_2 \Rightarrow ? \to ?\rangle\!\rangle; (? \to ?)!$$

$$\langle\!\langle G_{11} \to G_{12} \Rightarrow G_{21} \to G_{22}\rangle\!\rangle = \langle\!\langle G_{21} \Rightarrow G_{11}\rangle\!\rangle \to \langle\!\langle G_{12} \Rightarrow G_{22}\rangle\!\rangle \qquad \langle\!\langle ? \Rightarrow \text{Ref } G\rangle\!\rangle = (\text{Ref } ?)?; \langle\!\langle \text{Ref } ? \Rightarrow \text{Ref } G\rangle\!\rangle$$

$$\langle\!\langle \text{Ref } G \Rightarrow ?\rangle\!\rangle = \langle\!\langle \text{Ref } G \Rightarrow \text{Ref } ?\rangle\!\rangle; (\text{Ref } ?)! \qquad \langle\!\langle \text{Ref } G_2 \Rightarrow \text{Ref } G_1\rangle\!\rangle = \text{Ref } \langle\!\langle G_2 \Rightarrow G_1\rangle\!\rangle\ \langle\!\langle G_1 \Rightarrow G_2\rangle\!\rangle$$

Fig. 11. Coercion insertion function.

has become more precise. If the evidence is some ground type $\langle R\rangle$, then the judgment is mapped into the check-and-untag coercion from $?$ to R, followed by the tagging coercion from R to $?$. If the evidence is a function type (resp. reference type), then the corresponding coercion is the check-and-untag coercion from $?$ to $? \to ?$ (resp. Ref $?$), followed by the corresponding coercion of the consistent judgment between $? \to ?$ and $? \to ?$ (resp. Ref $? \sim$ Ref $?$) using the same evidence,[7] finally followed by the tagging coercion from $? \to ?$ (resp. Ref $?$) to $?$.

The bisimulation relation is formally presented in Fig. 13. This relation syntactically relates a $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ term and an HCC$^+$ term. Rules (b*const*) and (bλ) are straightforward. Rules (b*x*) and (b*b*) relate two variables and two constants respectively, where for simplicity we assume that x^G and x refer to the same source variable x, and b and b refer to the same constant b. Similarly, Rule (b*o*) relates two locations, assuming that the creation of locations is deterministic, i.e. new references in two related executions are always allocated at the same address: o^G and o refer to the same location o. Rule (bapp) relates two application terms inductively, but notice that because evidence terms do not correspond to anything in HCC$^+$, we build an ascribed term instead, e.g. to relate $\varepsilon_1 t_{11}$ with $\mathbf{t_{21}}$ we notice that the type of $\mathbf{t_{21}}$ has to be $G_1 \to G_2$, therefore as $\varepsilon \vdash G \sim G_1 \to G_2$ where $t_{11} \in \mathbb{T}[G]$, we can inductively relate $\varepsilon_1 t_{11} :: G_1 \to G_2$ with $\mathbf{t_{21}}$ instead. Similarly, we relate $\varepsilon_2 t_{12} :: G_1$ (as $\varepsilon_2 \vdash G' \sim G_1$, where $t_2 \in \mathbb{T}[G']$) with $\mathbf{t_2}$. We use the same reasoning for rules (bref), (b!), and (b:=). Rules (b::eq), (b::id), and (b::leq) are the most important rules. Rule (b::eq) is the most intuitive rule; it relates an ascribed term with a coerced term, only if the underlying evidence of the ascription is mapped to the coercion. Rule (b::id) relates a redundant ascription with a term without a coercion. The reason is that a term like $\iota_G \mathbf{u}$ (which is related to $\langle G\rangle u^G$ by (b::eq) if $\mathbf{u}$ is related to $u^G :: G$) reduces to $\mathbf{u}$, whereas in $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ this redundant cast is not eliminated. Rule (b::leq) relates $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ terms with HCC$^+$ terms that have eagerly combined coercions starting from the outermost pair of coercions. For instance,

[7] Note that if $\langle G_1 \to G_2\rangle \vdash ? \sim ?$ then $\langle G_1 \to G_2\rangle \vdash ? \to ? \sim ? \to ?$.

$$
\begin{aligned}
\llparenthesis \langle R\rangle \vdash R \sim R \rrparenthesis &= \iota_{\mathbf{R}}\\
\llparenthesis \langle ?\rangle \vdash ? \sim ? \rrparenthesis &= \iota_{?}\\
\llparenthesis \langle R\rangle \vdash ? \sim R \rrparenthesis &= \mathbf{R}?\\
\llparenthesis \langle R\rangle \vdash R \sim ? \rrparenthesis &= \mathbf{R}!\\
\llparenthesis \langle R\rangle \vdash ? \sim ? \rrparenthesis &= \mathbf{R}?;\mathbf{R}!\\
\llparenthesis \langle G_1 \to G_2\rangle \vdash G_{11} \to G_{12} \sim G_{21} \to G_{22} \rrparenthesis &= \llparenthesis \langle G_1\rangle \vdash G_{21} \sim G_{11} \rrparenthesis \to \llparenthesis \langle G_2\rangle \vdash G_{12} \sim G_{22} \rrparenthesis\\
\llparenthesis \langle G_1 \to G_2\rangle \vdash ? \sim G_{21} \to G_{22} \rrparenthesis &= (? \to ?)?; \llparenthesis \langle G_1 \to G_2\rangle \vdash ? \to ? \sim G_{21} \to G_{22} \rrparenthesis\\
\llparenthesis \langle G_1 \to G_2\rangle \vdash ? \sim ? \rrparenthesis &= (? \to ?)?; \llparenthesis \langle G_1 \to G_2\rangle \vdash ? \to ? \sim ? \to ? \rrparenthesis; (? \to ?)!\\
\llparenthesis \langle G_1 \to G_2\rangle \vdash G_{11} \to G_{12} \sim ? \rrparenthesis &= \llparenthesis \langle G_1 \to G_2\rangle \vdash G_{11} \to G_{12} \sim ? \to ? \rrparenthesis; (? \to ?)!\\
\llparenthesis \langle \mathsf{Ref}\ G\rangle \vdash \mathsf{Ref}\ G_1 \sim \mathsf{Ref}\ G_2 \rrparenthesis &= \mathsf{Ref}\ \llparenthesis \langle G\rangle \vdash G_2 \sim G_1 \rrparenthesis\ \llparenthesis \langle G\rangle \vdash G_1 \sim G_2 \rrparenthesis\\
\llparenthesis \langle \mathsf{Ref}\ G_1\rangle \vdash ? \sim \mathsf{Ref}\ G_2 \rrparenthesis &= (\mathsf{Ref}\ ?)?; \llparenthesis \langle \mathsf{Ref}\ G_1\rangle \vdash \mathsf{Ref}\ ? \sim \mathsf{Ref}\ G_2 \rrparenthesis\\
\llparenthesis \langle \mathsf{Ref}\ G_1\rangle \vdash ? \sim ? \rrparenthesis &= (\mathsf{Ref}\ ?)?; \llparenthesis \langle \mathsf{Ref}\ G_1\rangle \vdash \mathsf{Ref}\ ? \sim \mathsf{Ref}\ ? \rrparenthesis; (\mathsf{Ref}\ ?)!\\
\llparenthesis \langle \mathsf{Ref}\ G_1\rangle \vdash \mathsf{Ref}\ G_2 \sim ? \rrparenthesis &= \llparenthesis \langle \mathsf{Ref}\ G_1\rangle \vdash \mathsf{Ref}\ G_2 \sim \mathsf{Ref}\ ? \rrparenthesis; (\mathsf{Ref}\ ?)!
\end{aligned}
$$

$$\text{where } (G_i \neq R_i)$$

Fig. 12. Map from evidence augmented consistent judgments to coercions.

$$
(\text{b}b)\ \frac{}{b \approx b}
\qquad
(\text{b}x)\ \frac{}{x^G \approx x}
\qquad
(\text{b}\lambda)\ \frac{t_1 \approx \mathbf{t_2}}{\lambda x^G.t_1 \approx \lambda x : G.\mathbf{t_2}}
\qquad
(\text{b}o)\ \frac{}{o^G \approx o}
$$

$$
(\text{bapp})\ \frac{\varepsilon_1 t_{11} :: G_1 \to G_2 \approx \mathbf{t_{21}} \qquad \varepsilon_2 t_{12} :: G_1 \approx \mathbf{t_{22}}}{\varepsilon_1 t_{11} \mathbin{@^{G_1 \to G_2}} \varepsilon_2 t_{12} \approx \mathbf{t_{21}}\ \mathbf{t_{22}}}
\qquad
(\text{b::eq})\ \frac{t^{G'} \approx \mathbf{t} \qquad \mathbf{c} = \llparenthesis \varepsilon \vdash G' \sim G \rrparenthesis}{\varepsilon t^{G'} :: G \approx \mathbf{ct}}
$$

$$
(\text{b::id})\ \frac{t^G \approx \mathbf{t} \qquad \varepsilon = \langle G\rangle}{\varepsilon t^G :: G \approx \mathbf{t}}
\qquad
(\text{b::leq})\ \frac{\begin{array}{c}\mathbf{c_1} \vdash G_1 \Rightarrow G_2 \qquad \mathbf{c_2} = \llparenthesis \varepsilon \vdash G_2 \sim G_3 \rrparenthesis\\ t^{G_2} \approx \mathbf{c_1 t} \qquad \mathbf{c_1};\mathbf{c_2} \longmapsto^* \mathbf{c} \qquad \mathsf{nm}\ \mathbf{c}\end{array}}{\varepsilon t^{G_2} :: G_3 \approx \mathbf{ct}}
\qquad
(\text{bref})\ \frac{\varepsilon t_1 :: G \approx \mathbf{t_2}}{\mathsf{ref}^G\ \varepsilon t_1 \approx \mathsf{ref}\ \mathbf{t_2}}
$$

$$
(\text{b!})\ \frac{\varepsilon t_1 :: \mathsf{Ref}\ G \approx \mathbf{t_2}}{!^G \varepsilon t_1 \approx !\mathbf{t_2}}
\qquad
(\text{b:=})\ \frac{\varepsilon_1 t_{11} :: \mathsf{Ref}\ G_3 \approx \mathbf{t_{21}} \qquad \varepsilon_2 t_{12} :: G_3 \approx \mathbf{t_{22}}}{\varepsilon_1 t_{11} :=^{G_3} \varepsilon_2 t_{12} \approx \mathbf{t_{21}}{:=}\mathbf{t_{22}}}
$$

$$
(\text{bif})\ \frac{\varepsilon_1 t^{G_1} :: \mathsf{Bool} \approx \mathbf{t_1} \qquad G = (G_2 \sqcap G_3) \qquad \varepsilon_2 t^{G_2} :: G \approx \mathbf{t_2} \qquad \varepsilon_3 t^{G_3} :: G \approx \mathbf{t_3}}{\mathsf{if}\ \varepsilon_1 t^{G_1}\ \mathsf{then}\ \varepsilon_2 t^{G_2}\ \mathsf{else}\ \varepsilon_3 t^{G_3} \approx \mathsf{if}\ \mathbf{t_1}\ \mathsf{then}\ \mathbf{t_2}\ \mathsf{else}\ \mathbf{t_3}}
$$

$$
(\text{b}\oplus)\ \frac{ty(\oplus) = B_1 \times B_2 \to B_3 \qquad \varepsilon_1 t_{11} :: B_1 \approx \mathbf{t_{21}} \qquad \varepsilon_1 t_{12} :: B_2 \approx \mathbf{t_{22}}}{\varepsilon_1 t_{11} \oplus \varepsilon_2 t_{12} \approx \mathbf{t_{21}}\ \oplus\ \mathbf{t_{22}}}
$$

$$
(\text{b}\mu)\ \frac{\forall o^G \in \mu_1, o \in \boldsymbol{\mu_2}, o^G \approx o \Rightarrow \mu_1(o^G) \approx \boldsymbol{\mu_2}(o)}{\mu_1 \approx \boldsymbol{\mu_2}}
$$

Fig. 13. Bisimulation relation between intrinsic terms and the terms of the coercion calculus.

$t = \varepsilon_1(\varepsilon_2 t_1 :: G_2) :: G_1$ may be related to $\mathbf{t} = \mathbf{c_1}(\mathbf{c_2 t_2})$, if $t_1 \approx \mathbf{t_2}$, $\mathbf{c_1} = \llparenthesis \varepsilon_1 \vdash G_2 \sim G_1 \rrparenthesis$, and $\mathbf{c_2} = \llparenthesis \varepsilon_2 \vdash G_3 \sim G_1 \rrparenthesis$ for some G_3. But $\mathbf{t}$ may take a step to $\mathbf{c_{21} t_2}$ where $\mathbf{c_2}; \mathbf{c_1} \longmapsto^* \mathbf{c_{21}}$ and nm $\mathbf{c_{21}}$. By using (b::leq) we can relate t and $\mathbf{c_{21} t_2}$, by decomposing $\mathbf{c_{21}}$ back into $\mathbf{c_1}$ and $\mathbf{c_2}$, as we know by (b::eq) that $\varepsilon_2 t_1 :: G_2$ is related to $\mathbf{c_2 t_2}$. Finally rule (bμ) relates two stores if for all related locations, their corresponding values in the stores are related.

We can now state the bisimulation lemma between $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ and HCC$^+$ as follows

Lemma 15 *(Weak bisimulation between $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ and HCC$^+$). If $t_1 \in \mathbb{T}[G], \cdot; \Sigma \vdash_H \mathbf{t_2} : G$, $\boldsymbol{\mu_2} \models \Sigma$, $\mu_1 \approx \boldsymbol{\mu_2}$, and $t_1 \approx \mathbf{t_2}$, then*

1. *If $t_1 \mid \mu_1 \longmapsto t_1' \mid \mu_1'$, then $\mathbf{t_2} \mid \boldsymbol{\mu_2} \longmapsto^* \mathbf{t_2'} \mid \boldsymbol{\mu_2'}$ such that $t_1' \approx \mathbf{t_2'}$ and $\mu_1' \approx \boldsymbol{\mu_2'}$.*
2. *If $\mathbf{t_2} \mid \boldsymbol{\mu_2} \longmapsto \mathbf{t_2''} \mid \boldsymbol{\mu_2''}$, then $\exists j, 0 \leq j \leq 2, \mathbf{t_2''} \mid \boldsymbol{\mu_2''} \longmapsto^j \mathbf{t_2'} \mid \boldsymbol{\mu_2'}$ and $t_1 \mid \mu_1 \longmapsto^* t_1' \mid \mu_1'$ such that $t_1' \approx \mathbf{t_2'}$ and $\mu_1' \approx \boldsymbol{\mu_2'}$.*

In words, if the $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ term takes a step, then the related HCC$^+$ can take some steps to a related term/store configuration. Conversely, after taking a step, a term of HCC$^+$ may have to take up to two extra steps in order to be related to a (future)

$F ::= \square \mid E \oplus et \mid ev \oplus E \mid E \,@^{G}\, et \mid ev \,@^{G}\, E \mid$ ~~$E :: G$~~ $\mid$
$\quad \mathsf{if}\ E\ \mathsf{then}\ et\ \mathsf{else}\ et \mid \mathsf{ref}^{G}\ E \mid !^{G} E \mid E :=^{G} et \mid ev :=^{G} E$
$E ::= F \mid \langle G\rangle F \mid \boxed{\square :: G}$

$$(r7) \qquad \langle G_2\rangle(\langle G_1\rangle\, t :: G) \longrightarrow_c \begin{cases} \langle G_1 \sqcap G_2\rangle\, t \\ \varepsilon_{\mathbf{err}} t & \text{if } G_1 \sqcap G_2 \text{is not defined} \end{cases}$$

$$(r8) \qquad \varepsilon_{\mathbf{err}}(\langle G_1\rangle t :: G) \longrightarrow_c \varepsilon_{\mathbf{err}} t$$

$$(r9) \qquad \varepsilon_{\mathbf{err}} u \longrightarrow_c \mathbf{error}$$

Fig. 14. $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$: Modifications for a space-efficient dynamic semantics.

$\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ term. This is because for cases like application and assignment, $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ does in one step what HCC^{+} may do in three. For instance, take the following two related terms: $(\varepsilon_1(\lambda x^{G}.t) \,@^{G_1 \to G_2}\, \varepsilon_2 u) \approx ((\mathbf{c_1} \to \mathbf{c_2})\,(\lambda x : G.t))\mathbf{c_3 u}$. After a step of reduction the two terms are not related; the HCC^{+} term has to take two more steps to relate the body of the lambdas (we ignore stores for simplicity):

$$(\varepsilon_1(\lambda x^{G}.t) \,@^{G_1\to G_2}\, \varepsilon_2 u) \longmapsto icod(\varepsilon_1)([(\varepsilon_2 \circ idom(\varepsilon_1)u :: G/x^{G}]t) :: G_2$$

$$((\mathbf{c_1} \to \mathbf{c_2})\,(\lambda x : G.\mathbf{t}))\mathbf{c_3 u} \longmapsto \mathbf{c_1}\ ((\lambda x : G.\mathbf{t})\ (\mathbf{c_2}\ \mathbf{c_3 u})) \longmapsto \mathbf{c_1}\ ((\lambda x : G.\mathbf{t})\ (\mathbf{c_3'u})) \longmapsto \mathbf{c_1}\ \mathbf{t}[\mathbf{c_3'u}/\mathbf{x}]$$

The key result is that given a $\lambda_{\widetilde{\text{REF}}}$ term, its elaboration to $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ and its translation to HCC^{+} are bisimilar.

Proposition 16 *(Translations are bisimilar). Given $t : G$, if $t \rightsquigarrow_n t_1 : G$, and $t \rightsquigarrow_c \mathbf{t_2} : G$, then $t_1 \approx \mathbf{t_2}$.*

A direct consequence of bisimilarity is that elaborating to $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ and translating to HCC^{+} yield programs that co-terminate, co-fail, or co-diverge. We write $t \Downarrow$ (resp. $t \Downarrow \mathbf{error}$), if $t \mid \cdot \longmapsto^{*} v \mid \mu$ (resp. $t \mid \cdot \longmapsto^{*} \mathbf{error}$) for some resulting store μ, and similarly, we write $\mathbf{t} \Downarrow$ (resp. $\mathbf{t} \Downarrow \mathbf{error}$), if $\mathbf{t} \mid \cdot \longmapsto^{*} \mathbf{v} \mid \mu$ (resp. $\mathbf{t} \mid \cdot \longmapsto^{*} \mathbf{error}$) for some resulting store μ.

Corollary 17. *Given $t : G$, if $t \rightsquigarrow_n t_1 : G$ and $t \rightsquigarrow_c \mathbf{t_2} : G$, then $t_1 \Downarrow \iff \mathbf{t_2} \Downarrow$ and $t_1 \Downarrow \mathbf{error} \iff \mathbf{t_2} \Downarrow \mathbf{error}$. (Co-divergence follows trivially.)*

5.3. *Recovering space efficiency in $\lambda_{\widetilde{REF}}$*

Although we have established that given a $\lambda_{\widetilde{\text{REF}}}$ term, its elaboration to $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ and its translation to HCC^{+} are bisimilar, the dynamic semantics of $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ are not space-efficient, *i.e.* ascriptions can be repeatedly accumulated during reduction as illustrated in §4.7. We now present the changes needed in the runtime semantics of $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ in order to enjoy a space-efficient operational semantics.

The main space efficiency problem with $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ is that the definition of evaluation contexts allows ascriptions and evidences to accumulate until the corresponding subterm is reduced to a value. Fig. 14 presents a space-efficient dynamic semantics variant with respect to the original dynamic semantics of Fig. 5 (changes are highlighted in gray). To achieve space efficiency, we eliminate the $E :: G$ evaluation context, so as to forbid reduction inside nested ascriptions. Instead of combining evidences starting from the innermost pair of evidences, rule $(r7)$ now combines evidence eagerly starting from the outermost pair of evidences by using the new $\boxed{\square :: G}$ evaluation context, before subterm t is reduced to a value.

Preserving the failure behavior. To preserve the failure behavior of the original dynamic semantics (and fail at the same point of execution), $(r7)$ cannot simply reduce to an error when consistent transitivity is not defined. For instance, consider $t = \langle \mathsf{Bool}\rangle(\langle \mathsf{Int}\rangle(\langle \mathsf{Int}\rangle 1 + \langle \mathsf{Int}\rangle 1) :: ?) :: ?$. Using the original dynamic semantics, this term reduces to an error as follows

$$t \mid \cdot \longmapsto \langle \mathsf{Bool}\rangle(\langle \mathsf{Int}\rangle 2 :: ?) :: ? \mid \cdot \longmapsto \mathbf{error}$$

If we combine evidences starting from the outermost pair of evidence, t would reduce to an error immediately, *i.e.* $t \mid \cdot \longmapsto \mathbf{error}$, because $\langle \mathsf{Int}\rangle \circ \langle \mathsf{Bool}\rangle$ is not defined.

If combination of evidences is not defined then instead of reducing directly to an error, we reduce to an evidence term using the *pending error evidence* $\varepsilon_{\mathbf{err}}$. The pending error evidence $\varepsilon_{\mathbf{err}}$ is defined such that $\varepsilon_{\mathbf{err}} \vdash G_1 \sim G_2$ for any G_1 and G_2. We also update the definition of intrinsic values v to raw values u, or ascribed simple values $\varepsilon u :: G$ where $\varepsilon \neq \varepsilon_{\mathbf{err}}$. Rule $(r8)$ just propagates evidence $\varepsilon_{\mathbf{err}}$ until rule $(r9)$ finally reduces a pending error evidence combined with a simple value to an error. Using the space-efficient semantics, t now reduces as follows:

$$t \mid \cdot \longmapsto \varepsilon_{\mathbf{err}}(\langle \mathsf{Int}\rangle 1 + \langle \mathsf{Int}\rangle 1) :: ? \longmapsto \varepsilon_{\mathbf{err}} 2 :: ? \longmapsto \mathbf{error}$$

Space efficiency. We now demonstrate that the *total cost* of maintaining evidences during reduction in $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ is bounded. We proceed analogously to Herman et al. [24], and refer to their work for more details.

First, to reason about the space required by evidences we introduce the notion of *size* and *height* of evidences and types:

$$\begin{aligned}
size(\langle G\rangle) &= size(G)\\
size(\varepsilon_{\mathbf{err}}) &= 1\\
size(?) &= 1\\
size(B) &= 1\\
size(G_1 \to G_2) &= 1 + size(G_1) + size(G_2)\\
size(\mathsf{Ref}\ G_1) &= 1 + size(G_1)
\end{aligned}$$

$$\begin{aligned}
height(\langle G\rangle) &= height(G)\\
height(\varepsilon_{\mathbf{err}}) &= 1\\
height(?) &= 1\\
height(B) &= 1\\
height(G_1 \to G_2) &= 1 + max(height(G_1), height(G_2))\\
height(\mathsf{Ref}\ G_1) &= 1 + height(G_1)
\end{aligned}$$

size and *height* respectively compute the size and depth of the abstract syntax of evidence and types. Note that the maximum number of children for a given node is two (functions), therefore we can bound the size of evidences in terms of its height as a binary tree:

Lemma 18. $\forall \varepsilon \vdash G_1 \sim G_2, size(\varepsilon) \leq 2^{height(\varepsilon)} - 1$

Note that we get a tighter bound than in HCC[8] because evidence composition in $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ is not part of the syntax of evidence. The height of any type computed by the meet operator is bounded by the maximum height of both types.

Lemma 19. *If* $G_1 \sqcap G_2 = G_3$*, then* $height(G_3) \leq max(height(G_1), height(G_2))$

This lemma allows us to establish two similar lemmas to bound the maximum height of evidences:

Lemma 20. *If* $\mathcal{I}_{=}(G_1, G_2) = \varepsilon$*, then* $height(\varepsilon) \leq max(height(G_1), height(G_2))$

Lemma 21. *If* $\varepsilon_1 \circ^{=} \varepsilon_2 = \varepsilon_3$*, then* $height(\varepsilon_3) \leq max(height(\varepsilon_1), height(\varepsilon_2))$

Given a $\lambda_{\widetilde{\text{REF}}}$ term and its elaboration to $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$, the size and height of every evidence found at any step of reduction is bounded by some types found during the elaboration.

Proposition 22. *If* $t \leadsto_n t : G$ *and* $t \mid \cdot \longmapsto^* t' \mid \mu'$ *such that* ε *occurs in* (t', μ')*, then there exists* G' *in the derivation of* $t \leadsto_n t : G$ *such that* $height(\varepsilon) \leq height(G')$ *and* $size(\varepsilon) \leq 2^{height(G')} - 1$.

Finally, we bound the total cost of maintaining evidences. To do this we define the size of a program configuration $p = \langle t, \mu\rangle$ as the sum of the sizes of its term and store subcomponents. Following [24], for the store, we only count the locations that an idealized garbage collector would consider live, by using an auxiliary *reachable* metafunction:

$$\begin{aligned}
size(\langle t, \mu\rangle) =& size(t) + size(\mu\ |_{reachable(t)})\\
size(\mu) =& \sum\nolimits_{o^G \in dom(\mu)} (size(o^G) + size(\mu(o^G)))\\
size(b) =&= 1\\
size(o^G) =& size(x^G) = 1 + size(G)
\end{aligned}$$

[8] In HCC the bound is $5(2^{height(\varepsilon)} - 1)$, where 5 represents the maximum width of a normalized coercion.

$$size(\lambda x^G.t) = 1 + size(x^G) + size(t)$$
$$size(\mathsf{ref}^{G_2}\ \varepsilon t^{G_1}) = size(!^{G_2}\varepsilon t^{G_1}) = size(\varepsilon t^{G_1} :: G_2) = 1 + size(\varepsilon) + size(t^{G_1}) + size(G_2)$$
$$size(\varepsilon_1 t^{G_1} :=^{G_3} \varepsilon_2 t^{G_2}) = size(\varepsilon_1 t^{G_1}\ @^{G_3}\ \varepsilon_2 t^{G_2}) = 1 + size(\varepsilon_1) + size(t^{G_1}) + size(\varepsilon_2) + size(t^{G_2}) + size(G_3)$$

We now compare the size of a program configuration with the size of a program configuration reduced in an "oracle" semantics, where evidences require no space. The oracular measure $size_{\mathrm{OR}}$ is defined analogous to *size*, but where $size_{\mathrm{OR}}(\varepsilon) = 0$.

Proposition 23. *If $t \rightsquigarrow_n t : G$ and $t \mid \cdot \longmapsto^* t' \mid \mu'$, then there exists G' in the derivation of $t \rightsquigarrow_n t : G$ such that $size(\langle t', \mu'\rangle) \in O(2^{height(G')} \cdot size_{\mathrm{OR}}(\langle t', \mu'\rangle))$.*

As explained by Herman et al. [24], this result shows that when reducing a term t, coercions occupy bounded space, which depends on the height of some type used in the type derivation of t.

Relating the space-efficient semantics and HCC$^+$. Regarding the result of §5.2, the new space-efficient semantics are now more closely related to HCC$^+$. In particular rule (b::leq) of Fig. 13 is not needed anymore as evidences and coercions are reduced in lock-step. The only difference between both semantics is that identity coercions are eliminated during reduction, whereas redundant evidences are not (and this is why we have to keep the (b::id) rule).

Eager space-efficient dynamic semantics. Alternatively, we could have defined the space-efficient dynamic semantics without rules ($r8$) and ($r9$), and where ($r7$) would be defined as follows:

$$\langle G_2\rangle(\langle G_1\rangle\, t :: G) \longrightarrow_c \begin{cases} \langle G_1 \sqcap G_2\rangle t & \\ \textbf{error} & \text{if } G_1 \sqcap G_2 \text{ is not defined}\end{cases}$$

This variant would yield a *more eager* semantics. Going back to the previous example where $t = \langle \mathsf{Bool}\rangle(\langle \mathsf{Int}\rangle(\langle \mathsf{Int}\rangle 1 + \langle \mathsf{Int}\rangle 1) :: ?) :: ?$, t would now reduce immediately to an error after trying to combine the $\langle \mathsf{Int}\rangle$ and $\langle \mathsf{Bool}\rangle$ outer evidences: $t \mid \cdot \longmapsto \textbf{error}$.

The main difference with the previous approach is that a program that may diverge using the original dynamic semantics may now also fail with an error (therefore the bisimulation would be weaker). To illustrate this, consider the following program, where Ω is a non-terminating term:

$$\langle \mathsf{Int}\rangle(\langle ?\rangle(\langle \mathsf{Bool}\rangle(\langle ?\rangle \Omega :: ?) :: \mathsf{Bool}) :: ?) :: \mathsf{Int}$$

Using the original dynamic semantics, this program diverges because Ω is evaluated first before combining the outer evidences. Using the first variant of the space-efficient semantics, this program also diverges because the outer evidence $\varepsilon_{\mathbf{err}}$ never triggers an error because Ω never reduces to a value. But using the eager variant of the space-efficient semantics, the program reduces to an error just after combining the $\langle \mathsf{Bool}\rangle$ and $\langle \mathsf{Int}\rangle$ evidences.

6. Encoding permissive and monotonic references in $\lambda_{\widetilde{\mathrm{REF}}}$

In this section we present $\lambda^{\mathrm{pm}}_{\widetilde{\mathrm{REF}}}$, an extension of $\lambda_{\widetilde{\mathrm{REF}}}$ with support for both permissive and monotonic references [39]. We codify permissive and monotonic references by introducing new term constructors for each form of reference in $\lambda^{\mathrm{pm}}_{\widetilde{\mathrm{REF}}}$. Encoding monotonic references is more difficult than encoding permissive references, as it involves extending the dynamic semantics of $\lambda^{\varepsilon}_{\widetilde{\mathrm{REF}}}$.

$\lambda^{\mathrm{pm}}_{\widetilde{\mathrm{REF}}}$ supports two constructors to create references: `ref` for guarded reference, and `mref` for monotonic references. For instance, to emulate the behavior of monotonic references in examples 3 and 5, we use `mref` as illustrated below:

```
let x = mref (4 :: ?)
let y: Ref Bool = x ← runtime error
y := true
!y
```

Example 3

```
let x = mref (4 :: ?)
let y: Ref Int = x
x := true ← runtime error
```

Example 5

6.1. Static semantics

We start by extending the syntax of $\lambda_{\widetilde{\mathrm{REF}}}$ as follows:

$$\begin{array}{llll} z & ::= & \mathsf{g} \mid \mathsf{p} \mid \mathsf{m} & \text{(reference mode)} \\ v & ::= & \ldots \mid o_z & \text{(values)} \\ t & ::= & \ldots \mid \mathsf{ref}^G_z\ t & \text{(terms)} \end{array}$$

$$\ldots \quad (Grefz)\frac{\boxed{z \in \{\mathsf{g},\mathsf{p}\}} \quad \Gamma; \Sigma \vdash t : G' \quad G' \sim G}{\Gamma; \Sigma \vdash \mathsf{ref}^{G}_{\boxed{z}}\ t : \mathsf{Ref}\ G} \qquad (Grefp)\frac{\Gamma; \Sigma \vdash t : G' \quad G' \sim \boxed{?}}{\Gamma; \Sigma \vdash \mathsf{ref}^{?}_{\boxed{\mathsf{p}}}\ t : \mathsf{Ref}\ \boxed{?}}$$

$$(Gl)\frac{o_{\boxed{z}} : G \in \Sigma}{\Gamma; \Sigma \vdash o_{\boxed{z}} : \mathsf{Ref}\ G}$$

Fig. 15. $\lambda^{\mathsf{pm}}_{\widetilde{\mathrm{REF}}}$: Extensions to the static semantics.

A *reference mode* z may be a guarded reference g, a permissive reference p, or a monotonic reference m (notice that the `mref` and `ref` constructors correspond to $\mathsf{ref}^{G}_{\mathsf{m}}$ and $\mathsf{ref}^{G}_{\mathsf{g}}$ respectively). Locations are now indexed by a reference mode o_z, e.g. o_{m} represent a monotonic reference. Reference terms are also indexed with a tag z to know which kind of reference to create during reduction.

Fig. 15 highlights the changes to the typing rules of Fig. 3. Rule $(G\mathit{ref})$ is split into $(G\mathit{refz})$ and $(G\mathit{refp})$, the former to type check guarded and monotonic references, and the latter for permissive references. Note that in $\lambda_{\widetilde{\mathrm{REF}}}$, references created at type $?$ ($\mathsf{Ref}\ ?$) already behave like permissive references. This is because stored values of locations typed $\mathsf{Ref}\ ?$ never change its type ($?$) allowing for any arbitrary update. Therefore, to support permissive references in $\lambda^{\mathsf{pm}}_{\widetilde{\mathrm{REF}}}$, we simply add the $(G\mathit{refp})$ type rule, which assigns type $\mathsf{Ref}\ ?$ to any permissive reference, as it may be used freely with any value of any type.

6.2. Dynamic semantics

Analogous to $\lambda_{\widetilde{\mathrm{REF}}}$, the dynamic semantics of $\lambda^{\mathsf{pm}}_{\widetilde{\mathrm{REF}}}$ are defined via elaboration to their intrinsic representation. The extended language of intrinsic terms is called $\lambda^{\mathsf{pm}\varepsilon}_{\widetilde{\mathrm{REF}}}$. The elaboration rules are identical to Fig. 6, save for terms and types corresponding to references, which are now indexed by a reference mode.

Fig. 16 presents selected rules of the dynamic semantics of $\lambda^{\mathsf{pm}\varepsilon}_{\widetilde{\mathrm{REF}}}$. We highlight in gray the key changes with respect to Fig. 5. Following Siek et al. [39], we use in some rules *evolving stores* to reduce programs. An evolving store ν is a mapping between locations and terms, and intuitively it represents a store with pending evidence combinations. Compared to Siek et al. [39], the fact that $\lambda^{\mathsf{pm}\varepsilon}_{\widetilde{\mathrm{REF}}}$ does not have pairs allows us to simplify the definition of an evolving store as a store with a *single* pending combination. Evolving stores are used to propagate ascriptions on monotonic locations recursively. Configurations are pairs of an intrinsic term and an evolving store. Rule $(r2)$ is factorized to perform the ascription of the argument separately, because the argument can be a monotonic reference, and thus the ascription needs to be propagated into the store. Rules $(r4)$, $(r5)$ and $(r6)$ are adapted by adding a reference mode z to the corresponding terms and types constructors. Rule $(r4)$, instead of reducing to a location, now reduces to an ascribed location. Although this ascription may seem redundant, it is used later by other rules to push more precise evidence information in the store when working with monotonic locations as shown in rule $(r7)$. Rule $(r6)$ is adapted for monotonic locations: instead of updating the location cell to a new value, the cell is updated to a new ascribed value with evidence information of what was before in that cell. By doing this, we make sure that the evidence of a cell can only gain precision. Rule $(r7)$ reduces an evidence term and a store, as the store may change during combination of evidence. There is also a new special case when the raw value u is a monotonic location. In that case, the underlying value in the store is ascribed with information of evidence $\langle G_3\rangle$ as it may gain precision (or fail!). Rule (r7) uses function ev which returns the outermost evidence of a term, and is defined as $ev(\varepsilon t :: G) = \varepsilon$. As there may be cycles in the store, this new ascription can trigger the same reduction again in the future. Following Siek et al. [39], to avoid infinite loops, this special case is considered only if we are gaining precision. But instead of demanding that $G' \neq \widetilde{tref}(G_3)$, to prove the dynamic gradual guarantee we have to impose a stronger condition: $G' \not\sqsubseteq \widetilde{tref}(G_3)$.

To illustrate rule $(r7)$, consider the following step of reduction:

$$\begin{aligned}&\langle \mathsf{Ref}\ (? \to \mathsf{Int})\rangle(\langle \mathsf{Ref}\ (? \to ?)\rangle o^{? \to ?}_{\mathsf{m}} :: \mathsf{Ref}\ (? \to ?)) \mid o^{? \to ?}_{\mathsf{m}} \mapsto (\langle ? \to ?\rangle(\lambda x : ?.x) :: ? \to ?)\\ \longrightarrow_c &\langle \mathsf{Ref}\ (? \to \mathsf{Int})\rangle o^{? \to ?}_{\mathsf{m}} \mid o^{? \to ?}_{\mathsf{m}} \mapsto (\langle \mathsf{Int} \to ?\rangle(\langle ? \to ?\rangle(\lambda x : ?.x) :: ? \to ?) :: ? \to ?)\end{aligned}$$

The corresponding value in the store of the monotonic location is updated to a new term: its evidence will gain precision when the evolving store is reduced to a store. Following Siek et al. [39], rules $(\mathrm{R}\nu)$ and $(\mathrm{R}\nu\mathrm{Err})$ are added to reduce evolving stores. Rule $(\mathrm{R}\nu\mathrm{Err})$ steps to an error when one of the terms in the store reduces to an error. Notice that, differently from Siek et al. [39], propagation of ascriptions stops when a non-monotonic location is encountered. In the example, the evolving store is reduced as follows:

$$\begin{aligned}&\langle \mathsf{Ref}\ (? \to \mathsf{Int})\rangle o^{? \to ?}_{\mathsf{m}} \mid o^{? \to ?}_{\mathsf{m}} \mapsto (\langle \mathsf{Int} \to ?\rangle(\langle ? \to ?\rangle(\lambda x : ?.x) :: ? \to ?) :: ? \to ?)\\ \longrightarrow_c &\langle \mathsf{Ref}\ (? \to \mathsf{Int})\rangle o^{? \to ?}_{\mathsf{m}} \mid o^{? \to ?}_{\mathsf{m}} \mapsto (\langle \mathsf{Int} \to ?\rangle(\lambda x : ?.x) :: ? \to ?)\end{aligned}$$

Finally, contexts $(\mathrm{R}F)$ and $(\mathrm{R}F\mathrm{err})$ are also adapted to include the store when combining evidences.

$$\begin{array}{rcl} & \dots & \\ \mu & ::= & \cdot \mid \mu, o^G_z \mapsto v \\ \nu & ::= & \cdot \mid \mu \mid \mu, o^G_z \mapsto et :: G, \mu \end{array}$$

Notions of Reduction

$$\textsc{Config}_G = \mathbb{T}[G] \times \textsc{EvolvingStore}$$

$$\dots$$

$$(r2)\quad (\langle G'_{11} \rightarrow G'_{12}\rangle(\lambda x^{G_{11}}.t))@^{G_1 \rightarrow G_2}\ (\langle G'_2\rangle u) \mid \mu \longrightarrow \begin{cases} \langle G'_{12}\rangle([v/x^{G_{11}}]t)\ :: G_2 \mid \nu \\ \textbf{error} \qquad \text{if } G'_2 \sqcap G'_{11} \text{ is not defined} \end{cases}$$
$$\text{where } \langle G'_{11}\rangle(\langle G'_2\rangle u :: G_1) :: G_{11} \mid \mu \longmapsto v \mid \nu$$

$$\dots$$

$$(r4)\quad \mathsf{ref}^{G_2}_z\ \langle G_1\rangle u \mid \mu \longrightarrow \langle \mathsf{Ref}\ G_2\rangle\, o^{G_2}_z :: \mathsf{Ref}\ G_2 \mid \mu[o^{G_2}_z \mapsto \langle G_1\rangle u :: G_2]$$
$$\text{where } o_z \notin dom(\mu)$$

$$(r5)\quad !^G(\langle \mathsf{Ref}\ G_1\rangle o^{G_2}_z) \mid \mu \longrightarrow \langle G_1\rangle v :: G \mid \mu \quad \text{where } v = \mu(o^{G_2}_z)$$

$$(r6)\quad \langle \mathsf{Ref}\ G_1\rangle o^G_z :=^{G_3} \langle G_2\rangle u \mid \mu \longrightarrow \begin{cases} \mathsf{unit} \mid \mu[o^G_z \mapsto t] \\ \textbf{error} \qquad \text{if } G_2 \sqcap G_3 \text{ is not defined} \end{cases}$$
$$\begin{array}{l}\text{where } \mu(o^G_z) = \langle G'\rangle u' :: G, t = \langle G_3\rangle(\langle G_2\rangle u :: G_3) :: G \text{ and} \\ \text{if } z = \mathsf{m} \text{ then } G_3 = G_1 \sqcap G',\ \text{otherwise } G_3 = G_1 \end{array}$$

$$(r7)\quad \langle G_2\rangle(\langle G_1\rangle u :: G) \mid \nu \longrightarrow_c \begin{cases} \langle G_3\rangle u \mid \nu \qquad \text{if } u \neq o^G_{\mathsf{m}} \\ \langle G_3\rangle u \mid \nu[u \mapsto \langle G_4\rangle \nu(u) :: G_5] \qquad \text{if } u = o^{G_5}_{\mathsf{m}}, G' \not\sqsubseteq \widetilde{tref}(G_3) \\ \textbf{error} \qquad \text{if } G_3 \text{ or } G_4 \text{ are not defined} \end{cases}$$
$$\text{where } ev(\nu(u)) = \langle G'\rangle, G_3 = G_1 \sqcap G_2,\ \text{and } G_4 = \widetilde{tref}(G_3) \sqcap G'$$

Reduction

$$(\mathrm{R}E)\ \frac{t^G_1 \mid \mu \longrightarrow t^G_2 \mid \nu}{E[t^G_1] \mid \mu \longmapsto E[t^G_2] \mid \nu} \qquad \dots \qquad (\mathrm{R}F)\ \frac{et \mid \mu \longrightarrow_c et' \mid \nu}{F[et] \mid \mu \longmapsto F[et'] \mid \nu} \qquad (\mathrm{R}F\mathrm{err})\ \frac{et \mid \mu \longrightarrow_c \textbf{error}}{F[et] \mid \mu \longmapsto \textbf{error}} \qquad \dots$$

$$(\mathrm{R}\nu)\ \frac{\nu(o^G_z) = et :: G \qquad et \mid \nu \longrightarrow_c et' \mid \nu'}{t^{G'} \mid \nu \longmapsto t^{G'} \mid \nu'[o^G_z \mapsto et' :: G]} \qquad (\mathrm{R}\nu\mathrm{Err})\ \frac{\nu(o^G_z) = et :: G \qquad et \mid \nu \longrightarrow_c \textbf{error}}{t^{G'} \mid \nu \longmapsto \textbf{error}}$$

Fig. 16. $\lambda^{\mathsf{pm}\varepsilon}_{\widetilde{\mathrm{REF}}}$: Dynamic semantics (selected rules).

6.3. Properties

$\lambda^{\mathsf{pm}}_{\widetilde{\mathrm{REF}}}$ satisfies all the properties described in §4.6. As ref is not part of the source language $\lambda^{\mathsf{pm}}_{\widetilde{\mathrm{REF}}}$, to make sense of the conservative extension of the static discipline properties (Propositions 11 and 58), any ref term must be converted to either a $\mathsf{ref}^G_{\mathsf{g}}$ or a $\mathsf{ref}^G_{\mathsf{m}}$ term (when considering fully precise programs, a guarded reference and a monotonic reference behave identically). Notice that converting a ref term to a $\mathsf{ref}^?_{\mathsf{p}}$ term breaks Propositions 11, e.g. $\vdash \mathsf{ref}\ 1 : \mathsf{Ref\ Int}$ but $\vdash \mathsf{ref}^?_{\mathsf{p}}\ 1 : \mathsf{Ref}\ ?$.

For monotonic references we state two properties that best describe their behavior.

Proposition 24 *(Monotonicity of the evolving heap). If $t^G \mid \nu \longmapsto t'^G \mid \nu'$, then $\forall o^{G'}_{\mathsf{m}} \in dom(\mu)$, $ev(\nu'(o^{G'}_{\mathsf{m}})) \sqsubseteq ev(\nu(o^{G'}_{\mathsf{m}}))$.*

Monotonicity of the evolving heap means that, by taking a step, for every monotonic reference the outermost evidence of the corresponding term in the heap may only gain precision.

Proposition 25 *(Monotonicity of the heap). If $t^G \mid \mu \longmapsto^* t'^G \mid \mu'$, then $\forall o^{G'}_{\mathsf{m}} \in dom(\mu)$, $\mu(o^{G'}_{\mathsf{m}}) = \varepsilon u :: G'$, then $\mu'(o^{G'}_{\mathsf{m}}) = \varepsilon' u' :: G'$ and $\varepsilon' \sqsubseteq \varepsilon$.*

Monotonicity of the heap states that by reducing a term, starting from a heap and ending from a heap, the evidence of the underlying value of a monotonic evidence may only gain precision.

To the best of our knowledge, the dynamic gradual guarantee has never been proven for monotonic references. We prove this result here, which turns out to be challenging, and requires subtle considerations. We start by defining some operations on intrinsic terms: *flat* is a partial function that combines every evidence of nested ascriptions (which we call *flattened evidence*), and *uval* extracts a simple value from a nested ascription or a value:

$$flat(\varepsilon t^{G} :: G) = \varepsilon \circ^{=} flat(t^{G}) \qquad flat(\varepsilon u :: G) = \varepsilon$$

$$uval(\varepsilon t^{G} :: G) = uval(t) \qquad uval(\varepsilon u :: G) = u$$

With these definitions we can now establish the precision relation between evolving stores.

$$(\sqsubseteq_{\nu})\;\frac{\begin{array}{c}\forall o^{G_1} \in dom(\nu_1).\exists o^{G_2} \in dom(\nu_2)\ s.t.\\ \Omega \vdash o^{G_1} \sqsubseteq o^{G_2} \quad G_1 \sqsubseteq G_2 \quad uval(\nu_1(o^{G_1})) \sqsubseteq uval(\nu_2(o^{G_2}))\\ flat(\nu_1(o^{G_1})) \text{ is defined} \Rightarrow flat(\nu_1(o^{G_1})) \sqsubseteq flat(\nu_2(o^{G_2}))\end{array}}{\Omega \vdash \nu_1 \sqsubseteq \nu_2}$$

The difference with respect to precision of regular stores is that the values in the evolving stores of the same location must satisfy that the simple values and the combination of nested evidences are related by precision. Note that if the simple values are related, but consistent transitivity is not defined for $flat(\nu_1(o^{G_1}))$, then the condition holds for that location. The intuition is that when defining the dynamic gradual guarantee, we need to relate evolving stores that may potentially fail in future steps. As we are only interested in the cases where the more precise term reduces, then the $flat(\nu_1(o^{G_1})) \sqsubseteq flat(\nu_2(o^{G_2}))$ requirement makes sense only when $flat(\nu_1(o^{G_1}))$ is defined. Note also that if both evolving stores are regular stores, then this definition coincides with the precision relation of stores defined for $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$.

We introduce the notion of *monotonic well-formedness*, written $\vdash_{\mathsf{m}} (t, \nu)$, which states that, for every monotonic location $\varepsilon o^{G}_{\mathsf{m}}$ found in either the term t or the evolving store ν, if the flattened evidence of the underlying value in the evolving heap $flat(\nu(o^{G}_{\mathsf{m}}))$ is defined, then $flat(\nu(o^{G}_{\mathsf{m}})) \sqsubseteq iref(\varepsilon)$. Intuitively, if a monotonic reference gains precision, then its underlying value must also gain precision (sometimes in future steps). Crucially, reduction preserves monotonic well-formedness:

Lemma 26 *(Monotonic well-formedness preservation). If $\vdash_{\mathsf{m}} (t, \nu)$ and $t \mid \nu \longmapsto t' \mid \nu'$, then $\vdash_{\mathsf{m}} (t', \nu')$.*

With the definitions of precision for evolving stores and monotonic well-formedness, we state the dynamic gradual guarantee as follows:

Proposition 27 *(Dynamic guarantee). Suppose $\vdash_{\mathsf{m}} (t_i, \nu_i)$, $t_1 \sqsubseteq t_2$ and $\nu_1 \sqsubseteq \nu_2$. If $t_1 \mid \nu_1 \longmapsto t_1' \mid \nu_1'$ then $t_2 \mid \nu_2 \longmapsto^{*} t_2' \mid \nu_2'$, such that $t_1' \sqsubseteq t_2'$ and $\nu_1' \sqsubseteq \nu_2'$.*

Note that the dynamic gradual guarantee is stated in an unusual way. First it requires that monotonic well-formedness holds for each related term. Second, as rule (r7) may reduce differently (subject to the type precision test, which may potentially endanger the dynamic gradual guarantee), both terms are not necessarily related after each step of reduction: one can reduce to an evolving store whereas the other does not. For this reason, we state the relation after taking zero or more steps for the less precise term and evolving store. The proof of the dynamic gradual guarantee differs from the proof of that property in $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ in the parts involving reduction of evolving stores. For these, the proof depends on the following two lemmas.

First, reducing the more precise evolving store preserves the precision relation:

Lemma 28. *Let $t_1 \mid \nu_1 \sqsubseteq t_2 \mid \nu_2$. If $t_1 \mid \nu_1 \longmapsto t_1 \mid \nu_1'$, then $\nu_1' \sqsubseteq \nu_2$.*

Second, we can safely reduce a less precise evolving store into a regular store:

Lemma 29. *If $\vdash_{\mathsf{m}} (t_i, \mu_i)$ and $t_1 \mid \mu_1 \sqsubseteq t_2 \mid \nu_2$ then $t_2 \mid \nu_2 \longmapsto^{*} t_2 \mid \mu_2$, such that $\mu_1 \sqsubseteq \mu_2$.*

This last property requires that no infinite cycles occur when reducing an evolving store. This fact depends on the monotonic well-formedness of terms and stores:

Lemma 30. *If $\vdash_{\mathsf{m}} (t, \mu[o^{G}_{\mathsf{m}} \mapsto \varepsilon_1(\varepsilon_2 u :: G) :: G])$, $t \mid \mu[o^{G}_{\mathsf{m}} \mapsto \varepsilon_1(\varepsilon_2 u :: G) :: G] \longmapsto t \mid \nu[o^{G}_{\mathsf{m}} \mapsto \varepsilon_3 u :: G]$, then $t \mid \nu[o^{G}_{\mathsf{m}} \mapsto \varepsilon_3 u :: G] \not\longmapsto^{*} t \mid \nu'[o^{G}_{\mathsf{m}} \mapsto \varepsilon_4(\varepsilon_5 u :: G) :: G]$.*

7. Related work

We have already extensively discussed the four main approaches to gradual references found in the literature [35,24,39]. Vitousek et al. [46] present Reticulated Python, a tool for experimenting with gradual typing in Python 3 with support for references. They give two different dynamic semantics for casts: guarded semantics (with support for guarded references), and *transient semantics*. Instead of performing proxying of function or wrapping of runtime values, the transient semantics translate source programs by inserting type checks at all elimination forms, and at the entry and output of function definitions. These checks only test if values *shallowly* conform to a given type: only immediately-checkable information is considered. Greenman and Felleisen [22] compare guarded and transient semantics, and show that soundness for the transient semantics is a weaker notion that only guarantees preservation of the top-level constructor of the static type of an expression. For instance, consider $h = (\lambda f : (? \rightarrow ?).f :: \mathsf{Int} \rightarrow \mathsf{Int})$, $g = (\lambda x : \mathsf{Bool}.x)$, and term $h\ g$. Using guarded semantics and either space efficient coercions, threesomes, or evidences based semantics, this program reduces to an error after reducing the body of h, because $\mathsf{Bool} \rightarrow \mathsf{Bool}$ (the type of the returned value, g) is not consistent with the expected return type, $\mathsf{Int} \rightarrow \mathsf{Int}$. On the contrary, using the transient semantics, this program reduces successfully to g as its conforms with the expected top-level type constructor (a function). Of course, if we evaluate the program $(h\ g)\ 1$ then we will get an error right after applying g with 1 (thanks to the check at the entry of the body of g). Similarly, checks involving pair types only test if a given value is a pair. In Reticulated Python, for references (objects) the story is slightly different. Checks involving reference types (object types) recursively inspect a given value. For instance, consider the previous application of $h\ g$ where now $h = (\lambda x : \mathsf{Ref}\ ?.x :: \mathsf{Ref}\ \mathsf{Int})$, and $g = \mathsf{ref}\ \mathsf{true}$. Using transient semantics this program reduces to an error as expected. After reducing the body of h, the resulting location content does not conform with the expected type $\mathsf{Ref}\ \mathsf{Int}$. But when we combine references and functions, the same issue as before manifests: if $h = (\lambda x : \mathsf{Ref}\ (? \rightarrow ?).x :: \mathsf{Ref}\ \mathsf{Int} \rightarrow \mathsf{Int})$, and $g = \mathsf{ref}\ (\lambda x : \mathsf{Bool}.x)$, then $h\ g$ reduces successfully to a location whose content conforms with the expected resulting type: a reference to a function. We believe that the discussion about transient semantics is orthogonal to references, and is extensively analyzed by Greenman and Felleisen [22], therefore we did not include it in the main body of this work.

Much prior work on gradual security typing also supports references [14,18,19,41], although imprecision is introduced exclusively via security labels, e.g. types like $\mathsf{Ref}\ \mathsf{Int}_?$ are supported but not types like $\mathsf{Ref}\ ?$. Toro et al. [41] derive their gradual security language using AGT. The semantics of references corresponds to guarded references in which imprecision is limited to security labels.

Cimini and Siek [10] present the Gradualizer, a methodology and algorithm to systematically derive the static and cast insertion semantics of a gradual language from a static type system. They illustrate the application of this methodology to a language with references. We conjecture that the resulting gradual language treats gradual references as guarded or monotonic references, but it is hard to know precisely as the dynamic semantics were left as future work. They later extend the Gradualizer to also be able to derive the dynamic semantics of a gradual language [11]. The handling of auxiliary structures such as the heap is however not supported. The authors mention informally how the algorithm could be adapted to references, and conjecture that the resulting dynamic semantics correspond to the guarded semantics of Herman et al. [24]; however the precise formal treatment of this extension is left as future work.

There are many languages that integrate static and dynamic typing in some way, and support references: TypeScript [31], Flow [16], Hack [15], Dart [13], and Typed Clojure [6]. These languages adopt another approach called *optional* typing [7], which allows programmers to partially introduce type annotations to capture some errors statically, but do not perform any additional runtime checks at runtime beyond those that are performed for fully-untyped programs. This means that the runtime semantics are not sound with respect to its static type system.

Finally, there are many efforts related to gradualizing advanced typing disciplines, such as typestates [47,21], ownership types [34], annotated type systems [40], effects [4,5,43], refinement types [30,29], parametric polymorphism [2,27,42,32], and the security type systems discussed above, among others. Since the formulation of the refined criteria for gradually-typed languages [38], however, only refinement types [30], data type refinements [29] and a non-standard polymorphic language with explicit sealing [32] have been shown to fully respect such guarantees. Toro et al. [41,42] reveal some tensions between semantic type-based properties (noninterference, parametricity) and the dynamic gradual guarantee when following a type-driven approach to gradual typing, as that induced by AGT among others. The present work contributes to the gradualization of advanced typing disciplines by deriving a gradual language with references that satisfies the refined criteria. Additionally, this work presents the first formal statement and proof of the conservative extension of the dynamic semantics of the static language for a gradual language derived using AGT. Finally, the proof of the dynamic gradual guarantee for monotonic references is also novel.

8. Conclusion

We present $\lambda_{\widetilde{\text{REF}}}$, a gradual language with support for mutable references. This language is derived step-by-step using the AGT methodology [20]. We compare the resulting language with other gradual languages with mutable references: monotonic references, permissive references, and guarded references. We find that $\lambda_{\widetilde{\text{REF}}}$ treats references as guarded references, similar to how references are treated in the coercion calculus of Herman et al. [24] (HCC).

We formalize this relation by introducing HCC$^+$, an adapted version of HCC with conditionals and binary operations. We prove semantic correspondence between $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ (the intrinsic semantics of $\lambda_{\widetilde{\text{REF}}}$) and HCC$^+$ for elaborated and translated

$\lambda_{\widetilde{\text{REF}}}$ terms. The main difference between both semantics has nothing to do with references, but with the order in which evidences/casts are combined. Under certain conditions (and contrary to HCC), a gradual language derived with AGT may accumulate an unbounded number of runtime checks. We describe the changes needed in the dynamic semantics of $\lambda_{\widetilde{\text{REF}}}$ to recover space efficiency. We also present $\lambda^{\text{pm}}_{\widetilde{\text{REF}}}$, an extension of $\lambda_{\widetilde{\text{REF}}}$ that supports both permissive and monotonic references. Finally, we formally prove that monotonic references satisfy the dynamic gradual guarantee, a non-trivial novel result that requires a careful consideration of updates to the store.

An interesting perspective for future work is to extend $\lambda_{\widetilde{\text{REF}}}$ with nominal subtyping. We anticipate that a refined interpretation of evidences (such as pair of type intervals) might be needed to precisely capture type bounds at runtime, similarly to the security label intervals used in evidence by Toro et al. [41]. It would also be interesting to port the space-efficiency technique developed here to other AGT-derived gradual languages.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Gradualizing λ_{REF}, elaborating $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$

In this section we present some proofs used in the gradualization of λ_{REF} and elaboration of $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$.

Proposition 31 *(Precision, inductively). The following inductive definition of type precision is equivalent to Definition 2.*

$$\frac{}{B \sqsubseteq B} \qquad \frac{G_1 \sqsubseteq G_1' \quad G_2 \sqsubseteq G_2'}{G_1 \to G_2 \sqsubseteq G_1' \to G_2'} \qquad \frac{G_1 \sqsubseteq G_2}{\text{Ref } G_1 \sqsubseteq \text{Ref } G_2} \qquad \frac{}{G \sqsubseteq ?}$$

Proof. We have to prove that $\gamma(G_1) \subseteq \gamma(G_2) \iff G_1 \sqsubseteq G_2$, where $G_1 \sqsubseteq G_2$ correspond to the inductive definition of type precision. We prove $\Rightarrow$ (the other direction is analogous). We proceed by induction on $\gamma(G_1) \subseteq \gamma(G_2)$.

Case *($\gamma(B) \subseteq \gamma(G_2)$).* If $G_2 = B$ then we have to prove that $\gamma(B) \subseteq \gamma(B) \Rightarrow B \sqsubseteq B$, which is trivial. If $G_2 = ?$ then we have to prove that $\gamma(B) \subseteq \text{TYPE} \Rightarrow B \sqsubseteq ?$, which is also trivial.

Case *($\gamma(G_{11} \to G_{12}) \subseteq \gamma(G_2)$).* If $\gamma(G_{11} \to G_{12}) = \{T_{11} \to T_{12} \mid T_{11} \in \gamma(G_{11}) \wedge T_{12} \in \gamma(G_{12})\}$, then G_2 is either $?$ and $\gamma(G_2) = \text{TYPE}$, but $G_1 \sqsubseteq G_2$ and the result holds, or G_2 is $G_{21} \to G_{22}$ such that $\gamma(G_{21} \to G_{22}) = \{T_{21} \to T_{22} \mid T_{21} \in \gamma(G_{21}) \wedge T_{22} \in \gamma(G_{22})\}$ and $\{T_{11} \to T_{12} \mid T_{11} \in \gamma(G_{11}) \wedge T_{12} \in \gamma(G_{12})\} \subseteq \{T_{21} \to T_{22} \mid T_{21} \in \gamma(G_{21}) \wedge T_{22} \in \gamma(G_{22})\}$. For this to be true then $\gamma(G_{11}) = \{T_{11} \in \gamma(G_{11})\} \subseteq \gamma(G_{21}) = \{T_{21} \in \gamma(G_{21})\}$, and $\gamma(G_{12}) = \{T_{12} \in \gamma(G_{12})\} \subseteq \gamma(G_{22}) = \{T_{22} \in \gamma(G_{22})\}$. By induction hypotheses on $\gamma(G_{11}) \sqsubseteq \gamma(G_{21})$ and $\gamma(G_{12}) \sqsubseteq \gamma(G_{22})$ we know that $G_{11} \sqsubseteq G_{21}$ and $G_{12} \sqsubseteq G_{22}$. Therefore $G_{11} \to G_{12} \sqsubseteq G_{21} \to G_{22}$ and the result holds.

Case *($\gamma(\text{Ref } G_{11}) \subseteq \gamma(G_2)$).* We proceed similar to case function. □

Proposition 32 *(Galois connection). $\langle \gamma, \alpha \rangle$ is a Galois connection,* i.e.*:*

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

Proof. We first proceed to prove a) by induction on the structure of the non-empty set S.

Case *($\{B\}$).* Then $\alpha(\{B\}) = B$. But $\gamma(B) = \{B\}$ and the result holds.

Case *($\{\overline{T_{i1} \to T_{i2}}\}$).* Then $\alpha(\{\overline{T_{i1} \to T_{i2}}\}) = \alpha(\{\overline{T_{i1}}\}) \to \alpha(\{\overline{T_{i2}}\})$. But by definition of γ, $\gamma(\alpha(\{\overline{T_{i1}}\}) \to \alpha(\{\overline{T_{i2}}\})) = \{T_1 \to T_2 \mid T_1 \in \gamma(\alpha(\{\overline{T_{i1}}\})), T_2 \in \gamma(\alpha(\{\overline{T_{i2}}\}))\}$. By induction hypotheses, $\{\overline{T_{i1}}\} \subseteq \gamma(\alpha(\{\overline{T_{i1}}\}))$ and $\{\overline{T_{i2}}\} \subseteq \gamma(\alpha(\{\overline{T_{i2}}\}))$, therefore $\{\overline{T_{i1} \to T_{i2}}\} \subseteq \{T_1 \to T_2 \mid T_1 \in \{\overline{T_{i1}}\}, T_2 \in \{\overline{T_{i2}}\}\} \subseteq \{T_1 \to T_2 \mid T_1 \in \gamma(\alpha(\{\overline{T_{i1}}\})), T_2 \in \gamma(\alpha(\{\overline{T_{i2}}\}))\}$ and the result holds.

Case *($\{\overline{\text{Ref } T_i}\}$).* Then $\alpha(\{\overline{\text{Ref } T_i}\}) = \text{Ref } \alpha(\{\overline{T_i}\})$. But by definition of γ, $\gamma(\text{Ref } \alpha(\{\overline{T_i}\})) = \{\text{Ref } T \mid T \in \gamma(\alpha(\{\overline{T_i}\}))\}$. By induction hypothesis, $\{\overline{T_i}\} \subseteq \gamma(\alpha(\{\overline{T_i}\}))$, therefore $\{\overline{\text{Ref } T_i}\} = \{\text{Ref } T \mid T \in \{\overline{T_i}\}\} \subseteq \{\text{Ref } T \mid T \in \gamma(\alpha(\{\overline{T_i}\}))\}$ and the result holds.

Case *($\{\overline{T_i}\}$ heterogeneous).* Then $\alpha(\{\overline{T_i}\}) = ?$ and therefore $\gamma(\alpha(\{\overline{T_i}\})) = \text{TYPE}$, 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 $\gamma(B) = \{B\}$, and $\alpha(\{B\}) = B$.

Case *($G_1 \rightarrow G_2$).* We have to prove that $\alpha(\gamma(G_1 \rightarrow G_2)) \sqsubseteq G_1 \rightarrow G_2$, which is equivalent to prove that $\gamma(\alpha(\widehat{T})) \subseteq \widehat{T}$, where $\widehat{T} = \gamma(G_1 \rightarrow G_2) = \{T_1 \rightarrow T_2 \mid T_1 \in \gamma(G_1), T_2 \in \gamma(G_2)\}$. Then $\widehat{T}$ has the form $\{\overline{T_{i1} \rightarrow T_{i2}}\}$, such that $\forall i, T_{i1} \in \gamma(G_1)$ and $T_{i2} \in \gamma(G_2)$. Also note that $\{\overline{T_{i1}}\} = \gamma(G_1)$ and $\{\overline{T_{i2}}\} = \gamma(G_2)$. But by definition of α, $\alpha(\{\overline{T_{i1} \rightarrow T_{i2}}\}) = \alpha(\{\overline{T_{i1}}\}) \rightarrow \alpha(\{\overline{T_{i2}}\})$ and therefore $\gamma(\alpha(\{\overline{T_{i1}}\}) \rightarrow \alpha(\{\overline{T_{i2}}\})) = \{T_1 \rightarrow T_2 \mid T_1 \in \gamma(\alpha(\{\overline{T_{i1}}\})), T_2 \in \gamma(\alpha(\{\overline{T_{i2}}\}))\}$. But by induction hypotheses $\gamma(\alpha(\{\overline{T_{i1}}\})) \subseteq \gamma(G_1)$ and $\gamma(\alpha(\{\overline{T_{i2}}\})) \subseteq \gamma(G_2)$ and the result holds.

Case *($\mathsf{Ref}\ G$).* We have to prove that $\alpha(\gamma(\mathsf{Ref}\ G)) \sqsubseteq \mathsf{Ref}\ G$, which is equivalent to prove that $\gamma(\alpha(\widehat{T})) \subseteq \widehat{T}$, where $\widehat{T} = \gamma(\mathsf{Ref}\ G) = \{\mathsf{Ref}\ T \mid T \in \gamma(G)\}$. Then $\widehat{T}$ has the form $\{\overline{\mathsf{Ref}\ T_i}\}$, such that $\forall i, T_i \in \gamma(G)$. Also note that $\{\overline{T_i}\} = \gamma(G)$. But by definition of α, $\alpha(\{\overline{\mathsf{Ref}\ T_i}\}) = \mathsf{Ref}\ \alpha(\{\overline{T_i}\})$ and therefore $\gamma(\mathsf{Ref}\ \alpha(\{\overline{T_i}\})) = \{\mathsf{Ref}\ T \mid T \in \gamma(\alpha(\{\overline{T_i}\}))\}$. But by induction hypothesis $\gamma(\alpha(\{\overline{T_i}\})) \subseteq \gamma(G)$ and the result holds.

Case *($?$).* Then we have to prove that $\gamma(\alpha(?)) \subseteq \gamma(?) = \textsc{Type}$, but this is always true and the result holds immediately. □

Proposition 33. $\widetilde{equate}(G_1, G_2) = G_1 \sqcap G_2$.
The meet operator is defined as $G_1 \sqcap G_2 = \alpha(\gamma(G_1) \cap \gamma(G_2))$, and inductively as:

$$B \sqcap B = B \qquad G_1 \sqcap G_2 = G_2 \sqcap G_1 \qquad G \sqcap ? = ? \sqcap G = G \qquad (G_{11} \rightarrow G_{12}) \sqcap (G_{21} \rightarrow G_{22}) = (G_{11} \sqcap G_{21}) \rightarrow (G_{12} \sqcap G_{22})$$

$$\mathsf{Ref}\ G_1 \sqcap \mathsf{Ref}\ G_2 = \mathsf{Ref}\ G_1 \sqcap G_2 \qquad G_1 \sqcap G_2 \textit{ is undefined otherwise}$$

Proof. By induction on G_1 and G_2. □

Proposition 34. *Let $P_1(T_1, T_2) \triangleq T_1 = dom(T_2)$. Then $\widetilde{P_1}(G_1, G_2) \iff G_1 \sim \widetilde{dom}(G_2)$.*

Proof. The $\Rightarrow$ direction by induction on $\widetilde{P_1}(G_1, G_2)$, and the $\Leftarrow$ direction by induction on $G_1 \sim \widetilde{dom}(G_2)$. □

Proposition 35. *Let $P_2(T_1, T_2) \triangleq T_1 = tref(T_2)$. Then $\widetilde{P_2}(G_1, G_2) \iff G_1 \sim \widetilde{tref}(G_2)$.*

Proof. The $\Rightarrow$ direction by induction on $\widetilde{P_2}(G_1, G_2)$, and the $\Leftarrow$ direction by induction on $G_1 \sim \widetilde{tref}(G_2)$. □

Proposition 36. *If $G_1 \sim G_2$, then $\mathcal{I}_=(G_1, G_2) = \langle G_1 \sqcap G_2, G_1 \sqcap G_2\rangle$.*

Proof. Notice that in this setting $\mathcal{I}_=(G_1, G_2) = \alpha(\{\langle T\rangle \mid T \in \gamma(G_1), T \in \gamma(G_2)\}) = \alpha(\{\langle T\rangle \mid T \in \gamma(G_1) \cap \gamma(G_2)\}) = \langle \alpha(\{\langle T\rangle \mid T \in \gamma(G_1) \cap \gamma(G_2)\})\rangle = \langle \alpha(\gamma(G_1) \cap \gamma(G_2)\})\rangle = \langle G_1 \sqcap G_2\rangle$. □

Lemma 37. $\langle G_1\rangle \circ^= \langle G_2\rangle = \langle G_1 \sqcap G_2, G_1 \sqcap G_2\rangle$.

Proof. Similar to Proposition 7. □

Proposition 38 *(Elaboration preserves typing). If $\Gamma; \Sigma \vdash t : G$ and $\Gamma; \Sigma \vdash t \rightsquigarrow_n t^G : G$, then $t^G \in \mathbb{T}[G]$.*

Proof. Straightforward induction on $\Gamma; \Sigma \vdash t : G$. We only present one case as the other are analogous.

Case *($\Gamma; \Sigma \vdash t_1 := t_2 : \mathsf{Unit}$).* We know by (Gasgn) that

$$(\text{Gasgn})\ \frac{\Gamma; \Sigma \vdash t_1 : G_1 \qquad \Gamma; \Sigma \vdash t_2 : G_2 \qquad G_2 \sim \widetilde{tref}(G_1)}{\Gamma; \Sigma \vdash t_1 := t_2 : \mathsf{Unit}}$$

Then by (TRasgn):

$$(T\text{Rasgn})\ \frac{\begin{array}{cc}\Gamma; \Sigma \vdash t_1 \rightsquigarrow_n t^{G_1} : G_1 & \Gamma; \Sigma \vdash t_2 \rightsquigarrow_n t^{G_2} : G_2 \\ G_3 = \widetilde{tref}(G_1) \quad \varepsilon_1 = \mathcal{I}_=(G_1, \mathsf{Ref}\ G_3) & \varepsilon_2 = \mathcal{I}_=(G_2, G_3)\end{array}}{\Gamma; \Sigma \vdash t_1 := t_2 \rightsquigarrow_n \varepsilon_1 t^{G_1} :=^{G_3} \varepsilon_2 t^{G_2} : \mathsf{Unit}}$$

By induction hypothesis on $\Gamma; \Sigma \vdash t_1 : G_1$, if $\Gamma; \Sigma \vdash t_1 \rightsquigarrow_n t^{G_1} : G_1$ then $t^{G_1} \in \mathbb{T}[G_1]$. Similarly by induction hypothesis on $\Gamma; \Sigma \vdash t_2 : G_2$, if $\Gamma; \Sigma \vdash t_2 \rightsquigarrow_n t^{G_2} : G_2$ then $t^{G_2} \in \mathbb{T}[G_2]$. Also by definition of the interior function, $\varepsilon_1 \vdash G_1 \sim \mathsf{Ref}\ G_3$ and $\varepsilon_2 \vdash G_2 \sim G_3$. Then by (IGasgn):

$$(\text{IGasgn})\frac{\begin{array}{cc} t^{G_1} \in \mathbb{T}[G_1] & \varepsilon_1 \vdash G_1 \sim \mathsf{Ref}\ G_3 \\ t^{G_2} \in \mathbb{T}[G_2] & \varepsilon_2 \vdash G_2 \sim G_3 \end{array}}{\varepsilon_1 t^{G_1} :=^{G_3} \varepsilon_2 t^{G_2} \in \mathbb{T}[\mathsf{Unit}]}$$

and the result holds. □

Appendix B. Type safety

In this section we present the proof of type safety for $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$.

Lemma 39 *(Canonical forms). Consider a value $v \in \mathbb{T}[G]$. Then either $v = u$, or $v = \varepsilon u :: G$ with $u \in \mathbb{T}[G']$ and $\varepsilon \vdash G' \sim G$. Furthermore:*

1. *If $G = \mathsf{Bool}$ then either $v = b$ or $v = \varepsilon b :: \mathsf{Bool}$ with $b \in \mathbb{T}[\mathsf{Bool}]$.*
2. *If $G = \mathsf{Int}$ then either $v = n$ or $v = \varepsilon n :: \mathsf{Int}$ with $n \in \mathbb{T}[\mathsf{Int}]$.*
3. *If $G = G_1 \longrightarrow G_2$ then either $v = (\lambda x^{G_1}.t^{G_2})$ with $t^{G_2} \in \mathbb{T}[G_2]$ or $v = \varepsilon(\lambda x^{G'_1}.t^{G'_2}) :: G_1 \longrightarrow G_2$ with $t^{G'_2} \in \mathbb{T}[G'_2]$ and $\varepsilon \vdash G'_1 \longrightarrow G'_2 \sim G_1 \longrightarrow G_2$.*
4. *If $G = \mathsf{Ref}\ G'$ then either $v = o^{G'}$ or $v = \varepsilon o^{G'}$ with $o^{G'} \in \mathbb{T}[\mathsf{Ref}\ G']$ and $\varepsilon \vdash \mathsf{Ref}\ G' \longrightarrow \mathsf{Ref}\ G$.*

Proof. By direct inspection of the formation rules of gradual intrinsic terms (Fig. 4). □

Lemma 40 *(Substitution). If $t^G \in \mathbb{T}[G]$ and $v \in \mathbb{T}[G_1]$, then $[v/x^{G_1}]t^G \in \mathbb{T}[G]$.*

Proof. By induction on the derivation of t^G. □

Proposition 41 *($\longrightarrow$ is well defined). If $t^G \vdash \mu'$ and $t^G \longrightarrow r$, then $r \in \textsc{Config}_G \cup \{\mathbf{error}\}$, and if $r = t'^G \mid \mu'$, then also $t'^G \vdash \mu'$ and $dom(\mu) \subseteq dom(\mu')$.*

Proof. By induction on the structure of a derivation of $t^G \longrightarrow r$, considering the last rule used in the derivation.

Case *(r1).* Then $t^G = t^{B_3} = \varepsilon_1 b_1 \oplus \varepsilon_2 b_2$. Then

$$(\text{IG}\oplus)\frac{b_1 \in \mathbb{T}[B_1] \quad \varepsilon_1 \vdash B_1 \sim B_1 \quad b_2 \in \mathbb{T}[B_2] \quad \varepsilon_2 \vdash B_2 \sim B_2 \quad ty(\oplus) = B_1 x B_2 \to B_3}{\varepsilon_1 b_1 \oplus \varepsilon_2 b_2 \in \mathbb{T}[B_3]}$$

Therefore

$$\varepsilon_1 b_1 \oplus \varepsilon_2 b_2 \mid \mu \longrightarrow b_3 \mid \mu \text{ where } b_3 = b_1 \llbracket \oplus \rrbracket\, b_2$$

But $b_3 \in \mathbb{T}[B_3]$ and the result holds.

Case *(r2).* Then $t^G = \varepsilon_1(\lambda x^{G_{11}}.t_1^{G12})\ @^{G_1 \to G_2}\ (\varepsilon_2 u)$ and $G = G_2$. Then

$$(\text{Iapp})\frac{\dfrac{\dfrac{\mathcal{D}_1}{t_1^{G12} \in \mathbb{T}[G_{12}]}}{(\lambda x^{G_{11}}.t_1^{G12}) \in \mathbb{T}[G_{11} \to G_{12}]} \quad \dfrac{\mathcal{D}_2}{u \in \mathbb{T}[G'_2]} \quad \varepsilon_2 \vdash G'_2 \sim G_1 \qquad \varepsilon_1 \vdash G_{11} \to G_{12} \sim G_1 \to G_2}{\varepsilon_1(\lambda x^{G_{11}}.t_1^{G12})\ @^{G_1 \to G_2}\ \varepsilon_2 u \in \mathbb{T}[G_2]}$$

If $\varepsilon' = (\varepsilon_2 \circ^{=} idom(\varepsilon_1))$ is not defined, then $t^G \longrightarrow \mathbf{error}$, and then the result hold immediately. Suppose that consistent transitivity does hold, then

$$\varepsilon_1(\lambda x^{G_{11}}.t_1^{G12})\ @^{G_1 \to G_2}\ \varepsilon_2 u \mid \mu \longrightarrow icod(\varepsilon_1)([(\varepsilon' u :: G_{11})/x^{G_{11}}]t) :: G_2 \mid \mu$$

As $\varepsilon_2 \vdash G'_2 \sim G_1$ and by inversion lemma $idom(\varepsilon_1) \vdash G_1 \sim G_{11}$, then $\varepsilon' \vdash G'_2 \sim G_{11}$. Therefore $\varepsilon' u :: G_{11} \in \mathbb{T}[G_{11}]$, and by Lemma 40, $t'^{G_{12}} = [(\varepsilon' u :: G_{11})/x^{G_{11}}]t^{G_{12}} \in \mathbb{T}[G_{12}]$.

Then

$$(\text{IG::})\,\frac{t'^{G_{12}} \in \mathbb{T}[G_{12}] \qquad icod(\varepsilon_1) \vdash G_{12} \sim G_2}{icod(\varepsilon_1)t'^{G_{12}} :: G_2 \in \mathbb{T}[G_2]}$$

and the result holds.

Case *(r3 — true).* Then $t^G = \text{if } \varepsilon_1 b \text{ then } \varepsilon_2 t^{G_2} \text{ else } \varepsilon_3 t^{G_3}$, $G = G_2 \sqcap G_3$ and

$$(\text{IGif})\,\frac{\begin{array}{cc} b \in \mathbb{T}[G_1] \quad \varepsilon_1 \vdash G_1 \sim \mathsf{Bool} & G = (G_2 \sqcap G_3) \\ t^{G_2} \in \mathbb{T}[G_2] & \varepsilon_2 \vdash G_2 \sim G \\ t^{G_3} \in \mathbb{T}[G_3] & \varepsilon_3 \vdash G_3 \sim G \end{array}}{\text{if } \varepsilon_1 b \text{ then } \varepsilon_2 t^{G_2} \text{ else } \varepsilon_3 t^{G_3} \in \mathbb{T}[G]}$$

Therefore

$$\text{if } \varepsilon_1 b \text{ then } \varepsilon_2 t^{G_2} \text{ else } \varepsilon_3 t^{G_3} \mid \mu \longrightarrow \varepsilon_2 t^{G_2} :: G_2 \sqcap G_3 \mid \mu$$

But

$$(\text{IG::})\,\frac{t^{G_2} \in \mathbb{T}[G_2] \qquad \varepsilon \vdash G_2 \sim G_2 \sqcap G_3}{\varepsilon_2 t^{G_2} :: G_2 \sqcap G_3 \in \mathbb{T}[G_2 \sqcap G_3]}$$

and the result holds.

Case *(r3 — false).* Analogous to case (if-true).

Case *(r4).* Then $t^G = \text{ref}^{G_2}\ \varepsilon u$. Then

$$(\text{IGref})\,\frac{u \in \mathbb{T}[G_1] \qquad \varepsilon \vdash G_1 \sim G_2}{\text{ref}^{G_2}\ \varepsilon u \in \mathbb{T}[\mathsf{Ref}\ G_2]}$$

Then

$$\text{ref}^{G_2}\ \varepsilon u \mid \mu \longrightarrow o^{G_2} \mid \mu[o^{G_2} \mapsto \varepsilon u :: G_2]$$

where $o \notin dom(\mu)$. But as $\varepsilon u :: G_2 \in \mathbb{T}[G_2]$, then $o^{G_2} \vdash \mu[o^{G_2} \mapsto \varepsilon u :: G_2]$. Also $o^{G_2} \in \mathbb{T}[\mathsf{Ref}\ G_2]$ and the result holds.

Case *(r5).* Then $t^G = !^{G_2}(\varepsilon o^{G_1})$. Then

$$(\text{IGasgn})\,\frac{o^{G_1} \in \mathbb{T}[\mathsf{Ref}\ G_1] \qquad \varepsilon \vdash \mathsf{Ref}\ G_1 \sim \mathsf{Ref}\ G_2}{!^{G_2}(\varepsilon o^{G_1}) \in \mathbb{T}[G_2]}$$

Then

$$!^{G_2}(\varepsilon o^{G_1}) \mid \mu \longrightarrow iref(\varepsilon)v :: G_2 \mid \mu$$

where $v = \mu(o^{G_1})$ As μ is well formed, then $v \in \mathbb{T}[G_1]$. Then by inversion lemma $iref(\varepsilon) \vdash G_1 \sim G_2$, therefore $iref(\varepsilon)v :: G_2 \in \mathbb{T}[G_2]$ and the result holds.

Case *(r6).* Then $t^G = \varepsilon_1 o^{G_1} :=^{G_3} \varepsilon_2 u$. Then

$$(\text{IGasgn})\,\frac{\begin{array}{cc} o^{G_1} \in \mathbb{T}[\mathsf{Ref}\ G_1] & \varepsilon_1 \vdash \mathsf{Ref}\ G_1 \sim \mathsf{Ref}\ G_3 \\ u \in \mathbb{T}[G_2] & \varepsilon_2 \vdash G_2 \sim G_3 \end{array}}{\varepsilon_1 o^{G_1} :=^{G_3} \varepsilon_2 u \in \mathbb{T}[\mathsf{Unit}]}$$

If $\varepsilon' = (\varepsilon_2 \sqcap iref(\varepsilon_1))$ is not defined, then $t^G \longrightarrow$ **error**, and then the result hold immediately. Suppose that consistent transitivity does hold, then

$$\varepsilon_1 o^{G_1} :=^{G_3} \varepsilon_2 u \mid \mu \longrightarrow \text{unit} \mid \mu[o^{G_1} \mapsto \varepsilon' u :: G_1]$$

As $\varepsilon_2 \vdash G_2 \sim G_3$ and by inversion lemma $iref(\varepsilon_1) \vdash G_1 \sim G_3$, and as evidence is symmetrical $iref(\varepsilon_1) \vdash G_3 \sim G_1$, then $\varepsilon' \vdash G_2 \sim G_1$. Therefore $\varepsilon' u :: G_1 \in \mathbb{T}[G_1]$, and therefore $\text{unit} \vdash \mu[o^{G_1} \mapsto \varepsilon' u :: G_1]$. Also

$$\frac{\theta(\text{unit}) = \mathsf{Unit}}{\text{unit} \in \mathbb{T}[\mathsf{Unit}]}$$

and the result holds. □

Proposition 42 *($\longmapsto$ is well defined). If $t^G \vdash \mu$ and $t^G \mid \mu \longmapsto r$, then $r \in \textsc{Config}_G \cup \{\mathbf{error}\}$, and if $r = t'^G \mid \mu'$, then also $t'^G \vdash \mu'$ and $dom(\mu) \subseteq dom(\mu')$.*

Proof. By induction on the structure of a derivation of $t^G \longmapsto r$.

Case *($R\longrightarrow$).* $t^G \mid \mu \longrightarrow r$. By well-definedness of $\longrightarrow$ (Proposition 41), $r \in \textsc{Config}_G \cup \{\mathbf{error}\}$, and if $r = t'^G \mid \mu'$, then also $t'^G \vdash \mu'$ and $dom(\mu) \subseteq dom(\mu')$.

Case *(RE).* $t^G = E[t_1^{G'}]$, $E[t_1^{G'}] \in \mathbb{T}[G]$, $t_1^{G'} \mid \mu \longmapsto t_2^{G'} \mid \mu'$, $t_1^{G'} \in \mathbb{T}[G']$, and $E : \mathbb{T}[G'] \to \mathbb{T}[G]$. By induction hypothesis, $t_2^{G'} \in \mathbb{T}[G']$, so $E[t_2^{G'}] \in \mathbb{T}[G]$.

By induction hypothesis we also know that $t_2^{G'} \vdash \mu'$.

If $freeLocs(t_2^{G'}) \subseteq \mu'$, $freeLocs(f[t_1^G]) \subseteq \mu$, and $dom(\mu) \subseteq dom(\mu')$, then it is easy to see that $freeLocs(f[t_2^{G'}]) \subseteq \mu'$, and therefore conclude that $f[t^{G_2}] \vdash \mu'$.

Case *(REerr, RFerr).* $r = \mathbf{error}$.

Case *(RF).* Let $\textsc{EvTerm}_{G_2}$ be notation for the family of evidence terms εt^{G_1} such that $\varepsilon \vdash G_1 \sim G_2$. Then $t^G = F[et]$, $F[et] \in \mathbb{T}[G]$, and $F : \textsc{EvTerm}_{G_x} \to \mathbb{T}[G]$, and $et \longrightarrow_c et'$. Then there exists G_e, G_x such that $et = \varepsilon_e t_e^{G_e}$ and $\varepsilon_e \vdash G_e \sim G_x$. Also, $t_e = \varepsilon_v u :: G_e$, with $u \in \mathbb{T}[G_v]$ and $\varepsilon_v \vdash G_v \sim G_e$.

We know that $\varepsilon_c = \varepsilon_v \circ^{=} \varepsilon_e$ is defined, and $et = \varepsilon_e t_e \longrightarrow_c \varepsilon_c u = et'$. By definition of $\circ^{=}$ we have $\varepsilon_c \vdash G_v \sim G_x$, so $F[et'] \in \mathbb{T}[G]$.

As $freeLocs(et) = freeLocs(et')$ and $\mu' = \mu$ then it is easy to conclude that $F[et'] \vdash \mu$. $\square$

Now we can establish type safety: programs do not get stuck, though they may terminate with cast errors. Also the store of a program is well typed.

Proposition 43 *(Type Safety). If $t^G \in \mathbb{T}[G]$ then one of the following is true:*

1. *t^G is a value v;*
2. *if $t^G \vdash \mu$ then $t^G \mid \mu \longmapsto t'^G \mid \mu'$ for some term $t'^G \in \mathbb{T}[G]$ and some μ' such that $t'^G \vdash \mu'$ and $dom(\mu) \subseteq dom(\mu')$;*
3. *$t^G \mid \mu \longmapsto$ **error**.*

Proof. By induction on the structure of t^G. We only present some cases as all proceed the same way.

Case *(IGc, IGx, IGλ, IGo).* t^G is a value.

Case *(IG ::).* $t^G = \varepsilon_1 t^{G_1} :: G_2$, and

$$(\text{I::})\,\frac{t^{G_1} \in \mathbb{T}[G_1] \qquad \varepsilon_1 \vdash G_1 \sim G_2}{\varepsilon_1 t^{G_1} :: G_2 \in \mathbb{T}[G_2]}$$

By induction hypothesis on t^{G_1}, one of the following holds:

1. t^{G_1} is a simple value u, in which case t^G is also a value.
2. t^{G_1} is an ascribed value v, then the result holds by Proposition 42 and either (RF), or (RFerr).
3. $t^{G_1} \mid \mu \longmapsto r_1$ for some $r_1 \in \textsc{Config}_G \cup \{\mathbf{error}\}$. Hence $t^G \mid \mu \longmapsto r$ for some $r \in \textsc{Config}_G \cup \{\mathbf{error}\}$ by Proposition 42 and either (RE), or (REerr).

Case *(IGif).* $t^G = \mathsf{if}\ \varepsilon_1 t^{G_1}\ \mathsf{then}\ \varepsilon_2 t^{G_2}\ \mathsf{else}\ \varepsilon_3 t^{G_3}$ and

$$(\text{IGif})\,\frac{\begin{array}{ccc} t^{G_1} \in \mathbb{T}[G_1] & \varepsilon_1 \vdash G_1 \sim \mathsf{Bool} & G = (G_2 \sqcap G_3) \\ t^{G_2} \in \mathbb{T}[G_2] & \varepsilon_2 \vdash G_2 \sim G & \\ t^{G_3} \in \mathbb{T}[G_3] & \varepsilon_3 \vdash G_3 \sim G & \end{array}}{\mathsf{if}\ \varepsilon_1 t^{G_1}\ \mathsf{then}\ \varepsilon_2 t^{G_2}\ \mathsf{else}\ \varepsilon_3 t^{G_3} \in \mathbb{T}[G]}$$

By induction hypothesis on t^{G_1}, one of the following holds:

1. t^{G_1} is a simple value u, then by ($R\longrightarrow$), $t^G \mid \mu \longmapsto r$ and $r \in \textsc{Config}_G \cup \{\mathbf{error}\}$ by Proposition 42.

$$(\text{Sb})\dfrac{b = b \qquad \theta(c) = B \qquad \mu_1 \approx \mu_2}{\langle b, \mu_1\rangle \approx \langle b, \mu_2\rangle : B} \qquad\qquad \text{So}\dfrac{o = o \qquad \mu_1 \approx \mu_2}{\langle o, \mu_1\rangle \approx \langle o^G, \mu_2\rangle : B}$$

$$(\text{S}\lambda)\dfrac{\begin{array}{c}\mu_1 \approx \mu_2 \qquad\qquad \forall v_1', v_2', \varepsilon_1, \varepsilon_2, \mu_1', \mu_2'.\\ \mu_1 \sqsubseteq \mu_1', \mu_2 \subseteq \mu_2', \langle v_1', \mu_2'\rangle \approx \langle v_2', \mu_2'\rangle : T_1,\\ \varepsilon_1 \vdash T_1 \to T_2 \sim T_1 \to T_2 \qquad \varepsilon_2 \vdash T_1 \sim T_1\\ \langle v_1\ v_1', \mu_1'\rangle \approx \langle \varepsilon_1 u_2 \ @^{T_1 \to T_2}\ \varepsilon_2 v_2, \mu_2'\rangle : T_2\end{array}}{\langle v_1, \mu_1\rangle \approx \langle u_2, \mu_2\rangle : T_1 \to T_2} \qquad (\text{S::})\dfrac{\begin{array}{c}\langle v_1, \mu_1\rangle \approx \langle u_2, \mu_2\rangle : T\\ \varepsilon \vdash T \ \sim T\end{array}}{\langle v_1, \mu_1\rangle \approx \langle \varepsilon u_2 :: T, \mu_2\rangle : T}$$

$$(\text{S}t)\dfrac{\mu_1 \approx \mu_2 \qquad (t \mid \mu_1 \longmapsto^* v_1 \mid \mu_1' \iff t^T \mid \mu_2 \longmapsto^* v_2 \mid \mu_2' \quad \text{s.t.}\ \langle t, \mu_1'\rangle \approx \langle v_2, \mu_2'\rangle : T)}{\langle t, \mu_1\rangle \approx \langle t^T, \mu_2\rangle : T}$$

$$(\text{S}\mu)\dfrac{\forall o \in dom(\mu_1), o^T \in dom(\mu_2), o = o, \langle \mu_1(o), \mu_1\rangle \approx \langle \mu_2(o^T), \mu_2\rangle : T}{\mu_1 \approx \mu_2}$$

Fig. C.17. Logical relation between λ_{REF} and $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$.

2. t^{G_1} is an ascribed value v, then, $\varepsilon_1 t^{G_1} \longrightarrow_c r'$ for some $r' \in \text{EvTerm}_{\text{Bool}} \cup \{\mathbf{error}\}$. Hence $t^G \mid \ \mu \longmapsto r$ for some $r \in \text{Config}_G \cup \{\mathbf{error}\}$ by Proposition 42 and either ($\text{R}F$), or ($\text{R}F$err).
3. $t^{G_1} \mid \ \mu \longmapsto r_1$ for some $r_1 \in \mathbb{T}[G_1] \cup \{\mathbf{error}\}$. Hence $t^G \mid \ \mu \longmapsto r$ for some $r \in \text{Config}_G \cup \{\mathbf{error}\}$ by Proposition 42 and either ($\text{R}E$), or ($\text{R}E$err).

Case *(IGapp).* $t^G = (\varepsilon_1 t^{G_1})\ @^{G_{11} \to G_{12}}\ (\varepsilon_2 t^{G_2})$

$$(\text{IGapp})\dfrac{\begin{array}{cc} t^{G_1} \in \mathbb{T}[G_1] & \varepsilon_1 \vdash G_1 \sim G_{11} \to G_{12}\\ t^{G_2} \in \mathbb{T}[G_2] & \varepsilon_2 \vdash G_2 \sim G_{11}\end{array}}{(\varepsilon_1 t^{G_1})\ @^{G_{11} \to G_{12}}\ (\varepsilon_2 t^{G_2}) \in \mathbb{T}[G_{12}]}$$

By induction hypothesis on t^{G_1}, one of the following holds:

1. t^{G_1} is a value $(\lambda x^{G_{11}'}.t^{G_{12}'})$ (by canonical forms Lemma 39), posing $G_1 = G_{11}' \longrightarrow G_{12}'$.
   Then by induction hypothesis on t^{G_2}, one of the following holds:
   (a) t^{G_2} is a simple value u, then by ($\text{R}\longrightarrow$), $t^G \mid \ \mu \longmapsto r$ and $r \in \text{Config}_G \cup \{\mathbf{error}\}$ by Proposition 42.
   (b) t^{G_2} is an ascribed value v, then, $\varepsilon_2 t^{G_2} \longrightarrow_c r'$ for some $r' \in \text{EvTerm}_{G_{11}} \cup \{\mathbf{error}\}$. Hence $t^G \mid \ \mu \longmapsto r$ for some $r \in \text{Config}_G \cup \{\mathbf{error}\}$ by Proposition 42 and either ($\text{R}F$), or ($\text{R}F$err).
   (c) $t^{G_2} \mid \ \mu \longmapsto r_2$ for some $r_2 \in \text{Config}_{G_2} \cup \{\mathbf{error}\}$. Hence $t^G \mid \ \mu \longmapsto r$ for some $r \in \text{Config}_G \cup \{\mathbf{error}\}$ by Proposition 42 and either ($\text{R}E$), or ($\text{R}E$err).
2. t^{G_1} is an ascribed value v, then, $\varepsilon_1 t^{G_1} \longrightarrow_c r'$ for some $r' \in \text{EvTerm}_{G_{11} \to G_{12}} \cup \{\mathbf{error}\}$. Hence $t^G \mid \ \mu \longmapsto r$ for some $r \in \text{Config}_G \cup \{\mathbf{error}\}$ by Proposition 42 and either ($\text{R}F$), or ($\text{R}F$err).
3. $t^{G_1} \mid \ \mu \longmapsto r_1$ for some $r_1 \in \text{Config}_{G_1} \cup \{\mathbf{error}\}$. Hence $t^G \mid \ \mu \longmapsto r$ for some $r \in \text{Config}_G \cup \{\mathbf{error}\}$ by Proposition 42 and either ($\text{R}E$), or ($\text{R}E$err).

Case. Other cases are similar to the app case. □

Appendix C. Gradual guarantee

In this section we present the proof of the conservative extensions of the static discipline and the static and the dynamic gradual guarantee.

C.1. Conservative extensions of the static discipline

Proposition 44 *(Equivalence for fully-annotated terms (statics)). For any $t \in \text{Term}$, $. \vdash_S t : T$ if and only if $t : T$*

Proof. By induction over the typing derivations. The proof is trivial because static types are given singleton meanings via concretization. □

The equivalence for the dynamics of fully-annotated terms is defined in terms of a logical relation between terms of the static language λ_{REF}, and $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$ terms. The logical relation is presented in Fig. C.17.

Definition 9 *(Related substitutions).* We say that tuples $\langle \sigma_1, \mu_1\rangle$ and $\langle \sigma_2, \mu_2\rangle$ are related under Γ and Σ, notation $\Gamma; \Sigma \vdash \langle \sigma_1, \mu_1\rangle \approx \langle \sigma_2, \mu_2\rangle$ if $\sigma_1 \models \Gamma$, $\sigma_2 \models \Gamma$, $\Sigma \vdash \mu_1$, $\Sigma \vdash \mu_2$, $\mu_1 \approx \mu_2$, and

$$\forall x \in dom(\Gamma), \langle \sigma_1(x), \mu_1\rangle \approx \langle \sigma_2(x), \mu_2\rangle : \Gamma(x)$$

Definition 10 *(Semantic equivalence).*

$$\Gamma; \Sigma \vdash t \approx t^T : T \iff \forall \sigma_1, \sigma_2, \mu_1, \mu_1, \Sigma \vdash \mu_1, \Sigma \vdash \mu_2, \Gamma; \Sigma \vdash \langle \sigma_1, \mu_1\rangle \approx \langle \sigma_2, \mu_2\rangle, \\ \text{we have } \langle \sigma_1(t), \mu_1\rangle \approx \langle \sigma_2(t^T), \mu_2\rangle : T$$

Proposition 45 *(Fundamental property). For any $t \in$ Term, $\Gamma; \Sigma \vdash_s t : T$, $\Gamma; \Sigma \vdash t \rightsquigarrow_n t^T : T$, then $\Gamma; \Sigma \vdash t \approx t^T : T$.*

Proof. By induction on the type derivation of t.

Case *(Tx).* Then $t = x$ and therefore

$$(\text{Tx})\frac{x : T \in \Gamma}{\Gamma; \Sigma \vdash_s x : T}$$

Then we have to prove that $x \approx x^T : T$. But the result follows directly by Proposition 48.

Case *(Tb).* Then $t = b$ and therefore

$$(\text{Tc})\frac{\theta(b) = B}{\Gamma; \Sigma \vdash_s b : B}$$

and $t^T = b$. The result follows by Proposition 47.

Case *$(Tapp)$.* Then $t = t_1\ t_2$ and $T = T_2$ where

$$(\text{Tapp})\frac{\Gamma; \Sigma \vdash_s t_1 : T_1 \to T_2 \qquad \Gamma; \Sigma \vdash_s t_2 : T_1 \qquad T_2 = dom(T_1)}{\Gamma; \Sigma \vdash_s t_1\ t_2 : T_2}$$

and

$$(\mathit{TR}\text{app})\frac{\begin{array}{c}\Gamma; \Sigma \vdash t_1 \rightsquigarrow_n t^{G_1} : T_1 \to T_2 \qquad \Gamma; \Sigma \vdash t_2 \rightsquigarrow_n t^{T_2} : T_1 \\ \varepsilon_1 = \langle T_1 \to T_2\rangle = \mathcal{I}_=(T_1 \to T_2, \widetilde{dom}(T_1 \to T_2) \to \widetilde{cod}(T_1 \to T_2)) \\ \varepsilon_2 = \langle T_1\rangle = \mathcal{I}_=(T_1, \widetilde{dom}(T_1 \to T_2))\end{array}}{\Gamma; \Sigma \vdash t_1\ t_2 \rightsquigarrow_n \varepsilon_1 t^{T_1 \to T_2} @^{T_1 \to T_2} \varepsilon_2 t^{T_1} : T_2}$$

We have to prove that $\Gamma; \Sigma \vdash t_1\ t_2 \approx \varepsilon_1 t^{T_1 \to T_2} @^{T_1 \to T_2} \varepsilon_2 t^{T_1} : T_2$. By induction hypotheses we know that $\Gamma; \Sigma \vdash t_1 \approx t^{T_1 \to T_2} : T_1 \to T_2$ and that $\Gamma; \Sigma \vdash t_2 \approx t^{T_1} : T_1$. The result follows directly by Proposition 49.

Case *(Top).* Then $t = t_1 \oplus t_2$ and $T = B_3$, where

$$(\text{Top})\frac{\begin{array}{c}\Gamma; \Sigma \vdash_s t_1 : B_1 \qquad \Gamma; \Sigma \vdash_s t_2 : B_2 \\ ty(\oplus) = B_1 \times B_2 \to B_3\end{array}}{\Gamma; \Sigma \vdash_s t_1 \oplus t_2 : B_3}$$

and

$$(\mathit{TR}\text{op})\frac{\begin{array}{c}\Gamma; \Sigma \vdash t_1 \rightsquigarrow_n t^{B_1} : B_1 \qquad \Gamma; \Sigma \vdash t_2 \rightsquigarrow_n t^{B_2} : B_2 \qquad ty(\oplus) = B_1 x B_2 \to B_3 \\ \varepsilon_1 = \langle B_1\rangle = \mathcal{I}_=(B_1, B_1) \qquad\qquad \varepsilon_2 = \langle B_2\rangle = \mathcal{I}_=(B_2, B_2)\end{array}}{\Gamma; \Sigma \vdash t_1 \oplus t_2 \rightsquigarrow_n \varepsilon_1 t^{B_1} \oplus \varepsilon_2 t^{B_2} : B_3}$$

We have to prove that $\Gamma; \Sigma \vdash t_1 \oplus t_2 \approx \varepsilon_1 t^{B_1} \oplus \varepsilon_2 t^{B_2} : T_2$. By induction hypotheses we know that $\Gamma; \Sigma \vdash t_1 \approx t^{B_1} : B_1$ and that $\Gamma; \Sigma \vdash t_2 \approx t^{B_2} : B_2$. The result follows directly by Proposition 50.

Case *(Tif).* Then $t = \text{if } t_1 \text{ then } t_2 \text{ else } t_3$, where

$$(\text{Tif})\frac{\Gamma; \Sigma \vdash_s t_1 : \mathsf{Bool} \qquad \Gamma; \Sigma \vdash_s t_2 : T \qquad \Gamma; \Sigma \vdash_s t_3 : T}{\Gamma; \Sigma \vdash_s \text{ if } t_1 \text{ then } t_2 \text{ else } t_3 : T}$$

and

$$(\mathit{TR}\text{if})\frac{\begin{array}{c}\Gamma; \Sigma \vdash t_1 \rightsquigarrow_n t_1^{\mathsf{Bool}} : \mathsf{Bool} \qquad \Gamma; \Sigma \vdash t_2 \rightsquigarrow_n t_2^T : T \qquad \Gamma; \Sigma \vdash t_3 \rightsquigarrow_n t_3^T : T \\ \varepsilon_1 = \langle \mathsf{Bool}\rangle = \mathcal{I}_=(\mathsf{Bool}, \mathsf{Bool}) \qquad \varepsilon = \langle T\rangle = \mathcal{I}_=(T, T) \qquad \varepsilon = \langle T\rangle = \mathcal{I}_=(T, T)\end{array}}{\Gamma; \Sigma \vdash \text{ if } t_1 \text{ then } t_2 \text{ else } t_3 \rightsquigarrow_n \text{if } \varepsilon_1 t_1^{\mathsf{Bool}} \text{ then } \varepsilon t_2^T \text{ else } \varepsilon t_3^T : T}$$

We have to prove that $\Gamma;\Sigma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \approx \text{if } \varepsilon_1 t_1^{\mathsf{Bool}} \text{ then } \varepsilon t_2^T \text{ else } \varepsilon t_3^T : T$. By induction hypotheses we know that $\Gamma;\Sigma \vdash t_1 \approx t_1^{\mathsf{Bool}} : \mathsf{Bool}$ and that $\Gamma;\Sigma \vdash t_2 \approx t_2^T : T$, and $\Gamma;\Sigma \vdash t_3 \approx t_3^T : T$. The result follows directly by Proposition 51.

Case *(Tλ)*. Then $t = (\lambda x : T_1.t')$ and $T = T_1 \to T_2$, and therefore

$$(\mathrm{T}\lambda)\frac{\Gamma, x : T_1; \Sigma \vdash_s t' : T_2}{\Gamma; \Sigma \vdash_s (\lambda x : T_1.t') : T_1 \to T_2}$$

and

$$(TR\lambda)\frac{\Gamma, x : T_1 \vdash t' \rightsquigarrow_n t^{T_2} : T_2}{\Gamma; \Sigma \vdash \lambda x : T_1.t' \rightsquigarrow_n \lambda x^{T_1}.t^{T_2} : T_1 \to T_2}$$

Then we have to prove that $\Gamma;\Sigma \vdash (\lambda x : T_1.t') \approx (\lambda x^{T_1}.t^{T_2}) : T_1 \to T_2$. By induction hypothesis we already know that $\Gamma, x : T_1; \Sigma \vdash t' \approx t^{T_2} : T_2$. But the result follows directly by Proposition 52.

Case *(T::)*. Then $t = t' :: T$

$$(\mathrm{T}::)\frac{\Gamma; \Sigma \vdash_s t' : T \qquad T = T}{\Gamma; \Sigma \vdash_s (t' :: T) : T}$$

and

$$(TR::)\frac{\Gamma; \Sigma \vdash t' \rightsquigarrow_n t'^T : T \qquad \varepsilon = \langle T\rangle = \mathcal{I}_=(T, T)}{\Gamma; \Sigma \vdash (t :: T) \rightsquigarrow_n (\varepsilon t'^T :: T) : T}$$

We have to prove that $\Gamma;\Sigma \vdash t' :: T \approx \varepsilon t'^T :: T : T$. By induction hypothesis we know that $\Gamma;\Sigma \vdash t' \approx t'^T : T$. The result follows directly by Proposition 53.

Case *(Tref)*. Then $t = \mathsf{ref}\ t'$ and $T = \mathsf{Ref}\ T$, where

$$(\mathrm{Tref})\frac{\Gamma; \Sigma \vdash_s t' : T}{\Gamma; \Sigma \vdash_s \mathsf{ref}\ t' : \mathsf{Ref}\ T}$$

and

$$(TR\mathrm{ref})\frac{\Gamma; \Sigma \vdash t' \rightsquigarrow_n t'^T : T \qquad \varepsilon = \langle T\rangle = \mathcal{I}_=(T, T)}{\Gamma; \Sigma \vdash \mathsf{ref}\ t' \rightsquigarrow_n \mathsf{ref}^G\ \varepsilon t'^T : \mathsf{Ref}\ T}$$

We have to prove that $\Gamma;\Sigma \vdash \mathsf{ref}\ t' \approx \mathsf{ref}^G\ \varepsilon t'^T : \mathsf{Ref}\ T$. By induction hypothesis we know that $\Gamma;\Sigma \vdash t' \approx t'^T : T$. The result follows directly by Proposition 54.

Case *(Tderef)*. Then $t = !t'$, where

$$(\mathrm{Tderef})\frac{\Gamma; \Sigma \vdash_s t' : \mathsf{Ref}\ T}{\Gamma; \Sigma \vdash_s !t' : T}$$

and

$$(TR\mathrm{deref})\frac{\Gamma; \Sigma \vdash t' \rightsquigarrow_n t'^{\mathsf{Ref}\ T} : \mathsf{Ref}\ T \qquad \varepsilon = \langle \mathsf{Ref}\ T\rangle \mathcal{I}_=(\mathsf{Ref}\ T, \mathsf{Ref}\ T)}{\Gamma; \Sigma \vdash !t' \rightsquigarrow_n !^T \varepsilon t'^{\mathsf{Ref}\ T} : T}$$

We have to prove that $\Gamma;\Sigma \vdash !t' \approx !^T \varepsilon t'^T : T$. By induction hypothesis we know that $\Gamma;\Sigma \vdash t' \approx t'^{\mathsf{Ref}\ T} : \mathsf{Ref}\ T$. The result follows directly by Proposition 55.

Case *(Tasgn)*. Then $t = t_1 := t_2$ and $T = \mathsf{Unit}$, where

$$(\mathrm{Tasgn})\frac{\Gamma; \Sigma \vdash_s t_1 : \mathsf{Ref}\ T \qquad \Gamma; \Sigma \vdash_s t_2 : T}{\Gamma; \Sigma \vdash_s t_1 := t_2 : \mathsf{Unit}}$$

and

$$(TR\mathrm{asgn})\frac{\begin{array}{cc}\Gamma; \Sigma \vdash t_1 \rightsquigarrow_n t^{\mathsf{Ref}\ T} : \mathsf{Ref}\ T & \Gamma; \Sigma \vdash t_2 \rightsquigarrow_n t^T : T \\ \varepsilon_1 = \langle \mathsf{Ref}\ T\rangle = \mathcal{I}_=(\mathsf{Ref}\ T, \mathsf{Ref}\ T) & \varepsilon_2 = \langle T\rangle = \mathcal{I}_=(T, T)\end{array}}{\Gamma; \Sigma \vdash t_1 := t_2 \rightsquigarrow_n \varepsilon_1 t^{\mathsf{Ref}\ T} :=^T \varepsilon_2 t^T : \mathsf{Unit}}$$

We have to prove that $\Gamma;\Sigma \vdash t_1 := t_2 \approx \varepsilon_1 t^{\mathsf{Ref}\ T} :=^T \varepsilon_2 t^T : \mathsf{Unit}$. By induction hypotheses we know that $\Gamma;\Sigma \vdash t_1 \approx t^{\mathsf{Ref}\ T} : \mathsf{Ref}\ T$ and that $\Gamma;\Sigma \vdash t_2 \approx t^T : T$. The result follows directly by Proposition 56.

Case *(To).* Then $t = o$ and $T = \mathsf{Ref}\ T$, where

$$(\mathrm{To})\frac{o : T \in \Sigma}{\Gamma; \Sigma \vdash_s o : \mathsf{Ref}\ T}$$

and

$$(TRl)\frac{o : G \in \Sigma}{\Gamma; \Sigma \vdash o \rightsquigarrow_n o^G : \mathsf{Ref}\ G}$$

Then we have to prove that $o \approx o^G : \mathsf{Ref}\ T$. But the result follows directly by Proposition 57. □

Lemma 46. *Consider $\langle t, \mu_1\rangle \approx \langle t^T, \mu_2\rangle :$ and $\mu_1' \approx \mu_2'$, such that $\mu_1 \subseteq \mu_1'$ and $\mu_2 \subseteq \mu_2'$, then $\langle t, \mu_1'\rangle \approx \langle t^T, \mu_2'\rangle :$.*

Proof. Direct as evolution of the store to related store does not alter the relation between values that not depend on new locations. □

Proposition 47 *(Compatibility Tb). If $b \in B$, then $\Gamma; \Sigma \vdash b \approx b : B$.*

Proof. Trivial as $b = b$. □

Proposition 48 *(Compatibility Tx). If $x : T \in \Gamma$, then $\Gamma; \Sigma \vdash x \approx x^T : T$.*

Proof. Consider arbitrary $\sigma_1, \sigma_2, \mu_1, \mu_2$, such that $\Gamma; \Sigma \vdash \langle\sigma_1, \mu_1\rangle \approx \langle\sigma_2, \mu_2\rangle$. We are required to show that:

$$\langle\sigma_1(x), \mu_1\rangle \approx \langle\sigma_2(x^T), \mu_2\rangle : T$$

which is immediately by the definition of $\Gamma; \Sigma \vdash \Gamma; \Sigma \vdash \langle\sigma_1, \mu_1\rangle \approx \langle\sigma_2, \mu_2\rangle$. □

Proposition 49 *(Compatibility Tapp). If $\Gamma; \Sigma \vdash t_1 \approx t^{T_1 \to T_2} : T_1 \to T_2$, $\Gamma; \Sigma \vdash t_2 \approx t^{T_1} : T_1$, $\varepsilon_1 \vdash T_1 \to T_2 \sim T_1 \to T_2$, $\varepsilon_2 \vdash T_1 \sim T_1$, then $\Gamma; \Sigma \vdash t_1\ t_2 \approx \varepsilon_1 t^{T_1 \to T_2} @^{T_1 \to T_2}\ \varepsilon_2 t^{T_1} : T_2$.*

Proof. Consider arbitrary $\sigma_1, \sigma_2, \mu_1, \mu_2$, such that $\Gamma; \Sigma \vdash \langle\sigma_1, \mu_1\rangle \approx \langle\sigma_2, \mu_2\rangle$. We are required to show that:

$$\langle\sigma_1(t_1\ t_2), \mu_1\rangle \approx \langle\sigma_2(\varepsilon_1 t^{T_1 \to T_2} @^{T_1 \to T_2}\ \varepsilon_2 t^{T_1}), \mu_2\rangle : T_1 \to T_2$$

which, by definition of substitution, is equivalent to prove that

$$\langle\sigma_1(t_1)\ \sigma_1(t_2), \mu_1\rangle \approx \langle\varepsilon_1\sigma_2(t^{T_1 \to T_2}) @^{T_1 \to T_2}\ \varepsilon_2\sigma_2(t^{T_1}), \mu_2\rangle : T_1 \to T_2$$

We instantiate $\Gamma; \Sigma \vdash t_1 \approx t^{T_1 \to T_2} : T_1 \to T_2$ with σ_1, σ_2 and arbitrary μ_1 and μ_2 such that $\Sigma \vdash \mu_1$ and $\Sigma \vdash \mu_2$. We know then that $\langle t_1, \mu_1\rangle \approx \langle t^{T_1 \to T_2}, \mu_2\rangle : T_1 \to T_2$. Then suppose $\sigma_1(t_1) \mid \ \mu_1 \longmapsto^* v_{11} \mid \mu_1'$ and $\sigma_2(t^{T_1 \to T_2}) \mid \ \mu_2 \longmapsto^* v_{21} \mid \mu_2'$ (otherwise the result holds immediately). We know that $\langle v_{11}, \mu_1'\rangle \approx \langle v_{21}, \mu_2'\rangle : T_1 \to T_2$. Similarly we instantiate $\Gamma; \Sigma \vdash t_2 \approx t^{T_1} : T_1$ with σ_1, σ_2, μ_1' and μ_2'. Notice $\mu_1 \subseteq \mu_1'$ ($\mu_2 \subseteq \mu_2'$ resp.), therefore $\Sigma \vdash \mu_1'$ ($\Sigma \vdash \mu_2'$ resp.). Then we know that $\langle t_2, \mu_1'\rangle \approx \langle t^{T_1}, \mu_2'\rangle : T_1$. Then suppose $\sigma_1(t_2) \mid \ \mu_1' \longmapsto^* v_{12} \mid \mu_1''$ and $\sigma_2(t^{T_1}) \mid \ \mu_2' \longmapsto^* v_{22} \mid \mu_2''$ (otherwise the result holds immediately). We know that $\langle v_{21}, \mu_1''\rangle \approx \langle v_{22}, \mu_2''\rangle : T_1$. Let us assume $v_{21} = u_{21}$ (the other case is analogous modulo one trivial step of reduction). Then the result holds by definition of related lambdas instantiating with $\varepsilon_1, \varepsilon_2, \mu_1'', \mu_2'', v_{12}$ and v_{22}. □

Proposition 50 *(Compatibility Top). If $\Gamma; \Sigma \vdash t_1 \approx t^{B_1} : B_1$, $\Gamma; \Sigma \vdash t_2 \approx t^{B_2} : B_2$, $\varepsilon_1 \vdash B_1 \sim B_1$, $\varepsilon_2 \vdash B_2 \sim B_2$, $ty(\oplus) = B_1 x B_2 \to B_3$, then $\Gamma; \Sigma \vdash t_1 \oplus; t_2 \approx \varepsilon_1 t^{B_1} \oplus \varepsilon_2 t^{B_2} : B_3$.*

Proof. Consider arbitrary $\sigma_1, \sigma_2, \mu_1, \mu_2$, such that $\Gamma; \Sigma \vdash \langle\sigma_1, \mu_1\rangle \approx \langle\sigma_2, \mu_2\rangle$. We are required to show that:

$$\langle\sigma_1(t_1 \oplus t_2), \mu_1\rangle \approx \langle\sigma_2(\varepsilon_1 t^{B_1} \oplus \varepsilon_2 t^{B_2}), \mu_2\rangle : B_3$$

which, by definition of substitution, is equivalent to prove that

$$\langle\sigma_1(t_1) \oplus \sigma_1(t_2), \mu_1\rangle \approx \langle\varepsilon_1\sigma_2(t^{B_1}) \oplus \varepsilon_2\sigma_2(t^{B_2}), \mu_2\rangle : B_3$$

We instantiate $\Gamma; \Sigma \vdash t_1 \approx t^{B_1} : B_1$ with σ_1, σ_2 and arbitrary μ_1 and μ_2 such that $\Sigma \vdash \mu_1$ and $\Sigma \vdash \mu_2$. We know then that $\langle t_1, \mu_1\rangle \approx \langle t^{B_1}, \mu_2\rangle : B_1$. Then suppose $\sigma_1(t_1) \mid \ \mu_1 \longmapsto^* v_{11} \mid \mu_1'$ and $\sigma_2(t^{B_1}) \mid \ \mu_2 \longmapsto^* v_{21} \mid \mu_2'$ (otherwise the result holds immediately). We know that $\langle v_{11}, \mu_1'\rangle \approx \langle v_{21}, \mu_2'\rangle : B_1$. Similarly we instantiate $\Gamma; \Sigma \vdash t_2 \approx t^{B_2} : B_2$ with σ_1, σ_2, μ_1' and μ_2'. Notice $\mu_1 \subseteq \mu_1'$ ($\mu_2 \subseteq \mu_2'$ resp.), therefore $\Sigma \vdash \mu_1'$ ($\Sigma \vdash \mu_2'$ resp.). Then we know that $\langle t_2, \mu_1'\rangle \approx \langle t^{B_2}, \mu_2'\rangle : B_2$. Then suppose $\sigma_1(t_2) \mid \mu'_1 \longmapsto^* v_{12} \mid \mu''_1$ and $\sigma_2(t^{B_2}) \mid \mu'_2 \longmapsto^* v_{22} \mid \mu''_2$ (otherwise the result holds immediately). We know that $\langle v_{21}, \mu''_1\rangle \approx \langle v_{22}, \mu''_2\rangle : B_2$. Let us assume $v_{21} = u_{21}$ and $v_{22} = u_{22}$ (the other cases are analogous modulo one or two trivial steps of reduction). Then $\sigma_1(t_1) \oplus \sigma_1(t_2) \mid \mu_1 \longmapsto^* v_{11} \llbracket\oplus\rrbracket v_{12} \mid \mu''_1$ and $\varepsilon_1\sigma_2(t^{B_1}) \oplus \varepsilon_2\sigma_2(t^{B_2}) \mid \mu_2 \longmapsto^* \varepsilon_1 u_{21} \llbracket\oplus\rrbracket \varepsilon_2 u_{22} \mid \mu''_2 \longmapsto u_{21} \llbracket\oplus\rrbracket u_{22}$. But as $v_{11} = u_{21}$ and $v_{12} = u_{22}$, then $v_{11} \llbracket\oplus\rrbracket v_{12} = u_{21} \llbracket\oplus\rrbracket u_{22}$ and the result holds. □

Proposition 51 *(Compatibility Tif). If* $\Gamma; \Sigma \vdash t_1 \approx t_1^{\mathsf{Bool}} : \mathsf{Bool}$, $\Gamma; \Sigma \vdash t_2 \approx t_2^T : T$, $\Gamma; \Sigma \vdash t_3 \approx t^T : T$, $\varepsilon_1 \vdash \mathsf{Bool} \sim \mathsf{Bool}$, $\varepsilon \vdash T \sim T$, $\Gamma; \Sigma \vdash \mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3 \approx \mathsf{if}\ \varepsilon_1 t_1^{\mathsf{Bool}}\ \mathsf{then}\ \varepsilon t_2^T\ \mathsf{else}\ \varepsilon t_3^T : T$.

Proof. Consider arbitrary $\sigma_1, \sigma_2, \mu_1, \mu_2$, such that $\Gamma; \Sigma \vdash \langle\sigma_1, \mu_1\rangle \approx \langle\sigma_2, \mu_2\rangle$. We are required to show that:

$$\langle\sigma_1(\mathsf{if}\ t_1\ \mathsf{then}\ t_2\ \mathsf{else}\ t_3), \mu_1\rangle \approx \langle\sigma_2(\mathsf{if}\ \varepsilon_1 t_1^{\mathsf{Bool}}\ \mathsf{then}\ \varepsilon t_2^T\ \mathsf{else}\ \varepsilon t_3^T), \mu_2\rangle : T$$

which, by definition of substitution, is equivalent to prove that

$$\langle\mathsf{if}\ \sigma_1(t_1)\ \mathsf{then}\ \sigma_1(t_2)\ \mathsf{else}\ \sigma_1(t_3), \mu_1\rangle \approx \langle\mathsf{if}\ \varepsilon_1\sigma_2(t_1^{\mathsf{Bool}})\ \mathsf{then}\ \varepsilon\sigma_2(t_2^T)\ \mathsf{else}\ \varepsilon\sigma_2(t_3^T), \mu_2\rangle : T$$

We instantiate $\Gamma; \Sigma \vdash t_1 \approx t_1^{\mathsf{Bool}} : \mathsf{Bool}$ with σ_1, σ_2 and arbitrary μ_1 and μ_2 such that $\Sigma \vdash \mu_1$ and $\Sigma \vdash \mu_2$. We know then that $\langle t_1, \mu_1\rangle \approx \langle t_1^{\mathsf{Bool}}, \mu_2\rangle : \mathsf{Bool}$. Then suppose $\sigma_1(t_1) \mid \mu_1 \longmapsto^* v_{11} \mid \mu'_1$ and $\sigma_2(t^{\mathsf{Bool}}) \mid \mu_2 \longmapsto^* v_{21} \mid \mu'_2$ (otherwise the result holds immediately). We know that $\langle v_{11}, \mu'_1\rangle \approx \langle v_{21}, \mu'_2\rangle : \mathsf{Bool}$. Let us assume $v_{21} = u_{21}$ (the other case is analogous modulo one trivial step of reduction). Also let us assume $v_{11} = \mathsf{true}$ (the false case is analogous), therefore as $\langle v_{11}, \mu'_1\rangle \approx \langle u_{21}, \mu'_2\rangle : \mathsf{Bool}$, $u_{21} = \mathsf{true}$ as well. Then $t_1 \mid \mu_1 \longmapsto^* t_2 \mid \mu'_1$ and $t^{\mathsf{Bool}} \mid \mu_2 \longmapsto^* \mathsf{if}\ \langle\mathsf{Bool}\rangle\mathsf{true}\ \mathsf{then}\ \varepsilon\sigma_2(t_2^T)\ \mathsf{else}\ \varepsilon\sigma_2(t_3^T) \mid \mu'_2 \longmapsto \langle T\rangle t_2^T :: T \mid \mu'_2$.

But by instantiating $\Gamma; \Sigma \vdash t_2 \approx t_2^T : T$ with $\sigma_1, \sigma_2, \mu'_1$, and μ'_2, then $\langle t_2, \mu'_1\rangle \approx \langle t_2^T, \mu'_2\rangle$:, and then $t_2 \mid \mu'_1 \longmapsto^* v_{12} \mid \mu''_1$ and $t_2^T \mid \mu'_2 \longmapsto^* v_{22} \mid \mu''_2$, and $\langle v_{12}, \mu''_1\rangle \approx \langle v_{22}, \mu''_2\rangle$:. Let us assume $v_{22} = u_{22}$ (the other case is analogous), then as $\langle v_{12}, \mu''_1\rangle \approx \langle\langle T\rangle u_{22} :: T, \mu''_2\rangle$: the result holds. □

Proposition 52 *(Compatibility $T\lambda$). If* $\Gamma, x : T_1; \Sigma \vdash t' \approx t^{T_2} : T_2$, *then* $\Gamma; \Sigma \vdash (\lambda x : T_1.t') \approx (\lambda x^{T_1}.t^{T_2}) : T_1 \to T_2$.

Proof. Consider arbitrary $\sigma_1, \sigma_2, \mu_1, \mu_2$, such that $\Gamma; \Sigma \vdash \langle\sigma_1, \mu_1\rangle \approx \langle\sigma_2, \mu_2\rangle$. We are required to show that:

$$\langle\sigma_1((\lambda x : T_1.t')), \mu_1\rangle \approx \langle\sigma_2((\lambda x^{T_1}.t^{T_2})), \mu_2\rangle : T_1 \to T_2$$

which, by definition of substitution, is equivalent to prove that

$$\langle(\lambda x : T_1.\sigma_1(t')), \mu_1\rangle \approx \langle(\lambda x^{T_1}.\sigma_2(t^{T_2})), \mu_2\rangle : T_1 \to T_2$$

Consider $v'_1, v'_2, \varepsilon_1, \varepsilon_2, \mu'_1, \mu'_2$ such that $\mu_1 \sqsubseteq \mu'_1, \mu_2 \subseteq \mu'_2$, $\langle v'_1, \mu'_2\rangle \approx \langle v'_2, \mu'_2\rangle : T_1$, $\varepsilon_1 = \langle T_1 \to T_2\rangle \vdash T_1 \to T_2 \sim T_1 \to T_2$, and $\varepsilon_2 = \langle T_1\rangle \vdash T_1 \sim T_1$. We have to prove that

$$\langle(\lambda x : T_1.\sigma_1(t'))\ v'_1, \mu'_1\rangle \approx \langle\varepsilon_1(\lambda x^{T_1}.\sigma_2(t^{T_2}))\ @^{T_1 \to T_2}\ \varepsilon_2 v_2, \mu'_2\rangle : T_2$$

Consider $v_2 = u_2$ (the other case is trivial because if $v_2 = \varepsilon'_2 u_2 :: T_1$, then as everything is static the only possibility is that $\varepsilon'_2 = \langle T_1\rangle$ and as $\varepsilon_2 = \langle T_1\rangle$, consistent transitivity does not fail). By taking one step of evaluation (where consistent transitivity trivially does not fail):

$$(\lambda x : T_1.\sigma_1(t'))\ v'_1 \mid \mu'_1 \longmapsto \sigma_1(t')[v'_1/x]$$

and

$$\varepsilon_1(\lambda x^{T_1}.\sigma_2(t^{T_2}))\ @^{T_1 \to T_2}\ \varepsilon_2 v_2 \mid \mu'_2 \longmapsto \langle T_2\rangle(\sigma_2(t^{T_2})[\langle T_1\rangle u_2 :: T_1/x^{T_1}]) :: T_2 \mid \mu'_2$$

Notice that $\sigma_1(t')[v'_1/x] = \sigma'_1(t')$, where $\sigma'_1 = \sigma_1[x \mapsto v'_1]$, and analogously $\sigma_2(t^{T_2})[\langle T_1\rangle u_2 :: T_1/x^{T_1}] = \sigma'_2(t^{T_2})$, where $\sigma'_2 = \sigma_2[x^{T_1} \mapsto \langle T_1\rangle u_2 :: T_1]$. Also, as

$$\frac{\langle v_1, \mu'_1\rangle \approx \langle u_2, \mu'_2\rangle : T_1}{\langle v_1, \mu'_1\rangle \approx \langle\langle T_1\rangle u_2 :: T_1, \mu'_2\rangle :}\ T_1$$

then $\Gamma, x : T_1; \Sigma \vdash \langle\sigma'_1, \mu_1\rangle \approx \langle\sigma'_2, \mu_2\rangle$. Then by instantiating premise $\Gamma, x : T_1; \Sigma \vdash t' \approx t^{T_2} : T_2$ with $\sigma'_1, \mu_1, \sigma'_2$, and μ_2, we know that $\sigma'_1(t') \mid \mu'_1 \longmapsto^* v''_1 \mid \mu''_1 \iff \sigma'_2(t^{T_2}) \mid \mu'_2 \longmapsto^* v''_2 \mid \mu''_2$, where $\langle v''_1, \mu''_1\rangle \approx \langle v''_2, \mu''_2\rangle : T_2$. If $v''_2 = \langle T_2\rangle u''_2 :: T_2$ (the other case is similar), $\langle T_2\rangle v''_2 :: T_2 \mid \mu'_2 \to \langle T_2\rangle u''_2 :: T_2 \mid \mu'_2$, but $\langle v''_1, \mu''_1\rangle \approx \langle\langle T_2\rangle u''_2 :: T_2, \mu''_2\rangle : T_2$, and the result holds. □

Proposition 53 *(Compatibility T ::). If* $\Gamma; \Sigma \vdash t \approx t^T : T$, $\varepsilon \vdash T \sim T$, *then* $\Gamma; \Sigma \vdash t :: T \approx \varepsilon t^T :: T : T$.

Proof. Consider arbitrary $\sigma_1, \sigma_2, \mu_1, \mu_2$, such that $\Gamma; \Sigma \vdash \langle\sigma_1, \mu_1\rangle \approx \langle\sigma_2, \mu_2\rangle$. We are required to show that:

$$\langle\sigma_1(t :: T), \mu_1\rangle \approx \langle\sigma_2(\varepsilon t^T :: T), \mu_2\rangle : T$$

which, by definition of substitution, is equivalent to prove that

$$\langle\sigma_1(t) :: T, \mu_1\rangle \approx \langle\varepsilon\sigma_2(t^T) :: T, \mu_2\rangle : T$$

We instantiate $\Gamma; \Sigma \vdash t \approx t^T : T$ with σ_1, σ_2 and arbitrary μ_1 and μ_2 such that $\Sigma \vdash \mu_1$ and $\Sigma \vdash \mu_2$. We know then that $\langle t, \mu_1\rangle \approx \langle t^T, \mu_2\rangle : T$. Then suppose $\sigma_1(t) \mid \mu_1 \longmapsto^* v_1 \mid \mu_1'$ and $\sigma_2(t^T) \mid \mu_2 \longmapsto^* v_2 \mid \mu_2'$ (otherwise the result holds immediately). We know that $\langle v_1, \mu_1'\rangle \approx \langle v_2, \mu_2'\rangle : T$. Let us assume $v_2 = u_2$ (the other case is analogous modulo one trivial step of reduction). Then $v_1 :: T \mid \mu_1' \longmapsto v_1 \mid \mu_1'$, so we have to prove that $\langle v_1, \mu_1'\rangle \approx \langle\langle T\rangle u_2 :: T, \mu_2'\rangle : T$, which holds because $\langle v_1, \mu_1'\rangle \approx \langle u_2, \mu_2'\rangle : T$. □

Proposition 54 *(Compatibility Tref). If* $\Gamma; \Sigma \vdash t \approx t^T : T$, $\varepsilon \vdash T \sim T$, *then* $\Gamma; \Sigma \vdash \mathsf{ref}\ t \approx \mathsf{ref}^T\ \varepsilon t^T : \mathsf{Ref}\ T$.

Proof. Consider arbitrary $\sigma_1, \sigma_2, \mu_1, \mu_2$, such that $\Gamma; \Sigma \vdash \langle\sigma_1, \mu_1\rangle \approx \langle\sigma_2, \mu_2\rangle$. We are required to show that:

$$\langle\sigma_1(\mathsf{ref}\ t), \mu_1\rangle \approx \langle\sigma_2(\mathsf{ref}^T\ \varepsilon t^T), \mu_2\rangle : T$$

which, by definition of substitution, is equivalent to prove that

$$\langle\mathsf{ref}\ \sigma_1(t), \mu_1\rangle \approx \langle\mathsf{ref}^T\ \varepsilon\sigma_2(t^T), \mu_2\rangle : T$$

We instantiate $\Gamma; \Sigma \vdash t \approx t^T : T$ with σ_1, σ_2 and arbitrary μ_1 and μ_2 such that $\Sigma \vdash \mu_1$ and $\Sigma \vdash \mu_2$. We know then that $\langle t, \mu_1\rangle \approx \langle t^T, \mu_2\rangle : T$. Then suppose $\sigma_1(t) \mid \mu_1 \longmapsto^* v_1 \mid \mu_1'$ and $\sigma_2(t^T) \mid \mu_2 \longmapsto^* v_2 \mid \mu_2'$ (otherwise the result holds immediately). We know that $\langle v_1, \mu_1'\rangle \approx \langle v_2, \mu_2'\rangle : T$. Let us assume $v_2 = u_2$ (the other case is analogous modulo one trivial step of reduction). Then $\mathsf{ref}\ v_1 \mid \mu_1' \longmapsto o \mid \mu_1'[o \mapsto v_1]$, and $\mathsf{ref}^T\ \varepsilon u_2 \mid \mu_2' \longmapsto o^T \mid \mu_2'[o^T \mapsto \varepsilon u_2 :: T]$. Let $\mu_1'' = [o \mapsto v_1]$ and $\mu_2'' = [o^T \mapsto \varepsilon u_2 :: T]$. By (S::), $\langle v_1, \mu_1'\rangle \approx \langle\varepsilon u_2 :: T, \mu_2'\rangle : T$, and by Lemma 46, $\langle v_1, \mu_1''\rangle \approx \langle\varepsilon u_2 :: T, \mu_2''\rangle : T$, then by (S$\mu$), $\mu_1'' \approx \mu_2''$, and as $o = o$, by (So), $\langle o, \mu_1''\rangle \approx \langle o^T, \mu_2''\rangle : T$ and the result holds. □

Proposition 55 *(Compatibility Tderef). If* $\Gamma; \Sigma \vdash t \approx t^{\mathsf{Ref}\ T} : \mathsf{Ref}\ T$, $\varepsilon \vdash \mathsf{Ref}\ T \sim \mathsf{Ref}\ T$, *then* $\Gamma; \Sigma \vdash !t \approx !^{\mathsf{Ref}\ T}\varepsilon t^{\mathsf{Ref}\ T} : T$.

Proof. Consider arbitrary $\sigma_1, \sigma_2, \mu_1, \mu_2$, such that $\Gamma; \Sigma \vdash \langle\sigma_1, \mu_1\rangle \approx \langle\sigma_2, \mu_2\rangle$. We are required to show that:

$$\langle\sigma_1(!t), \mu_1\rangle \approx \langle\sigma_2(!^T\varepsilon t^{\mathsf{Ref}\ T}), \mu_2\rangle : T$$

which, by definition of substitution, is equivalent to prove that

$$\langle !\sigma_1(t), \mu_1\rangle \approx \langle !^T\varepsilon\sigma_2(t^{\mathsf{Ref}\ T}), \mu_2\rangle : T$$

We instantiate $\Gamma; \Sigma \vdash t \approx t^{\mathsf{Ref}\ T} : \mathsf{Ref}\ T$ with σ_1, σ_2 and arbitrary μ_1 and μ_2 such that $\Sigma \vdash \mu_1$ and $\Sigma \vdash \mu_2$. We know then that $\langle t, \mu_1\rangle \approx \langle t^{\mathsf{Ref}\ T}, \mu_2\rangle : \mathsf{Ref}\ T$. Then suppose $\sigma_1(t) \mid \mu_1 \longmapsto^* v_1 \mid \mu_1'$ and $\sigma_2(t^{\mathsf{Ref}\ T}) \mid \mu_2 \longmapsto^* v_2 \mid \mu_2'$ (otherwise the result holds immediately). We know that $\langle v_1, \mu_1'\rangle \approx \langle v_2, \mu_2'\rangle : \mathsf{Ref}\ T$. Let us assume $v_1 = o$ and $v_2 = o^T$ (the other case is analogous modulo one trivial step of reduction). Then $!v_1 \mid \mu_1' \longmapsto \mu_1'(o) \mid \mu_1'$, and $!^T\langle\mathsf{Ref}\ T\rangle u_2 \mid \mu_2' \longmapsto \langle T\rangle\mu_2'(o^T) :: T \mid \mu_2'$. By definition of $\mu_1' \approx \mu_2'$, $\langle\mu_1'(o), \mu_1'\rangle \approx \langle\mu_2'(o^T), \mu_2'\rangle : T$, therefore the result follows by (S::). □

Proposition 56 *(Compatibility Tasgn). If* $\Gamma; \Sigma \vdash t_1 \approx t^{\mathsf{Ref}\ T} : \mathsf{Ref}\ T$, $\Gamma; \Sigma \vdash t_2 \approx t^T : T$, $\varepsilon_1 \vdash \mathsf{Ref}\ T \sim \mathsf{Ref}\ T$, $\varepsilon_2 \vdash T \sim T$, *then* $\Gamma; \Sigma \vdash t_1{:=}t_2 \approx \varepsilon_1 t^{\mathsf{Ref}\ T} :=^T \varepsilon_2 t^T : \mathsf{Unit}$.

Proof. Consider arbitrary $\sigma_1, \sigma_2, \mu_1, \mu_2$, such that $\Gamma; \Sigma \vdash \langle\sigma_1, \mu_1\rangle \approx \langle\sigma_2, \mu_2\rangle$. We are required to show that:

$$\langle\sigma_1(t_1{:=}t_2), \mu_1\rangle \approx \langle\sigma_2(\varepsilon_1 t^{\mathsf{Ref}\ T} :=^T \varepsilon_2 t^T), \mu_2\rangle : \mathsf{Unit}$$

which, by definition of substitution, is equivalent to prove that

$$\langle\sigma_1(t_1){:=}\sigma_1(t_2), \mu_1\rangle \approx \langle\varepsilon_1\sigma_2(t^{\mathsf{Ref}\ T}) :=^T \varepsilon_2\sigma_2(t^T), \mu_2\rangle : \mathsf{Unit}$$

We instantiate $\Gamma; \Sigma \vdash t_1 \approx t^{\mathsf{Ref}\ T} : \mathsf{Ref}\ T$ with σ_1, σ_2 and arbitrary μ_1 and μ_2 such that $\Sigma \vdash \mu_1$ and $\Sigma \vdash \mu_2$. We know then that $\langle t_1, \mu_1\rangle \approx \langle t^{\mathsf{Ref}\ T}, \mu_2\rangle : \mathsf{Ref}\ T$. Then suppose $\sigma_1(t_1) \mid \mu_1 \longmapsto^* v_{11} \mid \mu_1'$ and $\sigma_2(t^{\mathsf{Ref}\ T}) \mid \mu_2 \longmapsto^* v_{21} \mid \mu_2'$ (otherwise the result holds immediately). We know that $\langle v_{11}, \mu_1'\rangle \approx \langle v_{21}, \mu_2'\rangle : \mathsf{Ref}\ T$. Similarly we instantiate $\Gamma; \Sigma \vdash t_2 \approx t^T : B_2$ with σ_1, σ_2, μ_1' and μ_2'. Notice $\mu_1 \subseteq \mu_1'$ ($\mu_2 \subseteq \mu_2'$ resp.), therefore $\Sigma \vdash \mu_1'$ ($\Sigma \vdash \mu_2'$ resp.). Then we know that $\langle t_2, \mu_1'\rangle \approx \langle t^{B_2}, \mu_2'\rangle : B_2$. Then suppose $\sigma_1(t_2) \mid \mu_1' \longmapsto^* v_{12} \mid \mu_1''$ and $\sigma_2(t^{B_2}) \mid \mu_2' \longmapsto^* v_{22} \mid \mu_2''$ (otherwise the result holds immediately). We know that $\langle v_{21}, \mu_1''\rangle \approx \langle v_{22}, \mu_2''\rangle : B_2$. Let us assume $v_{21} = u_{21} = o^T$ and $v_{22} = u_{22}$ (the other cases are analogous modulo one or two trivial steps of reduction). Then $\sigma_1(t_1){:=}\sigma_1(t_2) \mid \mu_1 \longmapsto^* o{:=}v_{12} \mid \mu_1'' \longmapsto \text{unit} \mid \mu_1''[o \mapsto v_{12}]$ and $\varepsilon_1\sigma_2(t^{\text{Ref }T}) :=^T \varepsilon_2\sigma_2(t^T) \mid \mu_2 \longmapsto^* \langle \text{Ref } T\rangle\sigma_2(o^T) :=^T \langle T\rangle u_{22} \mid \mu_2'' \longmapsto \text{unit} \mid \mu_2''[o^T \mapsto \langle T\rangle u_{22} :: T]$.

Let $\mu_1''' = [o \mapsto v_{12}]$ and $\mu_2''' = [o^T \mapsto \langle T\rangle u_{22} :: T]$. By (S::), $\langle v_{12}, \mu_1''\rangle \approx \langle\langle T\rangle u_{22} :: T, \mu_2''\rangle : T$, and by Lemma 46, $\langle v_1, \mu_1'''\rangle \approx \langle\langle T\rangle u_{22} :: T, \mu_2'''\rangle : T$, then by ($S\mu$), $\mu_1''' \approx \mu_2'''$, and as unit = unit, by (Sb), $\langle \text{unit}, \mu_1'''\rangle \approx \langle \text{unit}, \mu_2'''\rangle : T$ and the result holds. □

Proposition 57 *(Compatibility To). If $o : T \in \Sigma$, then $\Gamma; \Sigma \vdash o \approx o^T : T$.*

Proof. Direct by definition of related stores. □

Proposition 58 *(Equivalence for fully-annotated terms (dynamics)). For any $t \in \text{Term}$, $. \vdash_s t : T$, $t \leadsto_n t^T : T$, then $t \mid \cdot \longmapsto_s^* v \mid \mu \iff t^T \mid \cdot \longmapsto^* v' \mid \mu'$, for some μ, μ' such that $\langle v, \mu\rangle \approx \langle v', \mu'\rangle : T$.*

Proof. As a special case of the fundamental Property 45 and the unfolding of related computations. □

C.2. Static gradual guarantee

Definition 11 *(Term precision).*

$$(\text{Px})\frac{}{x \sqsubseteq x} \qquad (\text{Pc})\frac{}{c \sqsubseteq c} \qquad (\text{P}\lambda)\frac{t \sqsubseteq t' \quad G_1 \sqsubseteq G_1'}{(\lambda x : G_1.t) \sqsubseteq (\lambda x : G_1'.t')} \qquad (\text{P}\oplus)\frac{t_1 \sqsubseteq t_1' \quad t_2 \sqsubseteq t_2'}{t_1 \oplus t_2 \sqsubseteq t_1' \oplus t_2'}$$

$$(\text{Papp})\frac{t_1 \sqsubseteq t_1' \quad t_2 \sqsubseteq t_2'}{t_1\ t_2 \sqsubseteq t_1'\ t_2'} \qquad (\text{Pif})\frac{t \sqsubseteq t \quad t_1 \sqsubseteq t_1' \quad t_2 \sqsubseteq t_2'}{\text{if } t \text{ then } t_1 \text{ else } t_2 \sqsubseteq \text{if } t' \text{ then } t_1' \text{ else } t_2'} \qquad (\text{P::})\frac{t \sqsubseteq t' \quad G \sqsubseteq G'}{t :: G \sqsubseteq t' :: G'}$$

$$(\text{Pref})\frac{t \sqsubseteq t' \quad G \sqsubseteq G'}{\text{ref}^G\ t \sqsubseteq \text{ref}^{G'}\ t'} \qquad (\text{Pderef})\frac{t \sqsubseteq t'}{!t \sqsubseteq !t'} \qquad (\text{Passign})\frac{t_1 \sqsubseteq t_1' \quad t_2 \sqsubseteq t_2'}{t_1 := t_2 \sqsubseteq t_1' := t_2'} \qquad (\text{Po})\frac{}{o \sqsubseteq o}$$

Definition 12 *(Type environment precision).*

$$\frac{}{\cdot \sqsubseteq \cdot} \qquad \frac{\Gamma \sqsubseteq \Gamma' \quad G \sqsubseteq G'}{\Gamma, x : G \sqsubseteq \Gamma', x : G'}$$

Definition 13 *(Store typing precision).*

$$\frac{}{\cdot \sqsubseteq \cdot} \qquad \frac{\Sigma \sqsubseteq \Sigma' \quad G \sqsubseteq G'}{\Sigma, o : G \sqsubseteq \Sigma', o : G'}$$

Lemma 59. *If $\Gamma; \Sigma \vdash t : G$, $\Gamma \sqsubseteq \Gamma'$ and $\Sigma \sqsubseteq \Sigma'$, then $\Gamma'; \Sigma' \vdash t : G'$ for some $G \sqsubseteq G'$.*

Proof. Simple induction on typing derivations. □

Lemma 60. *If $G_1 \sim G_2$ and $G_1 \sqsubseteq G_1'$ and $G_2 \sqsubseteq G_2'$ then $G_1' \sim G_2'$.*

Proof. By definition of $\sim$, there exists $\langle T_1, T_2\rangle \in \langle \widehat{T}_1, \widehat{T}_2\rangle \in \gamma^2(G_1, G_2)$ such that $T_1 = T_2$. $G_1 \sqsubseteq G_1'$ and $G_2 \sqsubseteq G_2'$ mean that $\gamma(G_1) \subseteq \gamma(G_1')$ and $\gamma(G_2) \subseteq \gamma(G_2')$, therefore $\langle T_1, T_2\rangle \in \langle \widehat{T}_1, \widehat{T}_2\rangle \in \gamma^2(G_1', G_2')$. □

Proposition 61 *(Static gradual guarantee). If $t_1 : G_1$ and $t_1 \sqsubseteq t_2$, then $t_2 : G_2$, for some G_2 such that $G_1 \sqsubseteq G_2$.*

Proof. We prove the property on opens terms instead of closed terms: If $\Gamma; \Sigma \vdash t_1 : G_1$ and $t_1 \sqsubseteq t_2$ then $\Gamma; \Sigma \vdash t_2 : G_2$ and $G_1 \sqsubseteq G_2$.

The proof proceeds by induction on the typing derivation.

Case *(Gx, Gb).* Trivial by definition of $\sqsubseteq$ using (Px), (Pb) respectively.

Case *(Gλ).* Then $t_1 = (\lambda x : G_1.t)$ and $G_1 = G_1' \to G_2'$. By $(G\lambda)$ we know that:

$$(G\lambda)\frac{\Gamma, x : G_1'; \Sigma \vdash t : G_2'}{\Gamma \vdash (\lambda x : G_1'.t) : G_1' \to G_2'} \tag{C.1}$$

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

$$(G\lambda)\frac{t \sqsubseteq t' \qquad G'_1 \sqsubseteq G''_1}{(\lambda x : G'_1.t) \sqsubseteq (\lambda x : G''_1.t')} \tag{C.2}$$

Using induction hypotheses on the premise of C.1, $\Gamma, x : G'_1; \Sigma' \vdash t' : G''_2$ with $G'_2 \sqsubseteq G''_2$. By Lemma 59, $\Gamma, x : G''_1 \vdash t' : G'''_2$ where $G''_2 \sqsubseteq G'''_2$. Then we can use rule $(G\lambda)$ to derive:

$$(G\lambda)\frac{\Gamma, x : G''_1; \Sigma' \vdash t' : G'''_2}{\Gamma; \Sigma' \vdash (\lambda x : G'_1.t') : G''_1 \to G'''_2}$$

Where $G_2 \sqsubseteq G''_2$. Using the premise of C.2 and the definition of type precision we can infer that

$$G'_1 \to G'_2 \sqsubseteq G''_1 \to G'''_2$$

and the result holds.

Case *(G⊕)*. Then $t_1 = t'_1 \oplus t'_2$ and $G_1 = \mathsf{Int}$. By $(G\oplus)$ we know that:

$$(\mathrm{T}\oplus)\frac{\Gamma; \Sigma \vdash t_1 : G_1 \qquad \Gamma; \Sigma \vdash t_2 : G_2 \qquad G_1 \sim B_1 \qquad G_2 \sim B_2}{\Gamma; \Sigma \vdash t_1 \oplus t_2 : B_3} \tag{C.3}$$

Consider t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = t''_1 \oplus t''_2$ and therefore

$$(\mathrm{P}\oplus)\frac{t'_1 \sqsubseteq t''_1 \qquad t'_2 \sqsubseteq t''_2}{t'_1 \oplus t'_2 \sqsubseteq t''_1 \oplus t''_2} \tag{C.4}$$

Using induction hypotheses on the premises of C.3, $\Gamma; \Sigma \vdash t''_1 : G'_1$ and $\Gamma; \Sigma \vdash t''_2 : G'_2$, where $G_1 \sqsubseteq G'_1$ and $G_2 \sqsubseteq G'_2$. By Lemma 60, $G'_1 \sim B_1$ and $G'_2 \sim B_2$. Therefore we can use rule $(G\oplus)$ to derive:

$$(\mathrm{T}\oplus)\frac{\Gamma'; \Sigma' \vdash t'_1 : G'_1 \qquad \Gamma'; \Sigma' \vdash t'_2 : G'_2 \qquad G'_1 \sim B_1 \qquad G'_2 \sim B_2}{\Gamma'; \Sigma' \vdash t'_1 \oplus t'_2 : B_3}$$

and the result holds.

Case *(Gapp)*. Then $t_1 = t'_1\, t'_2$ and $G_1 = G_{12}$. By $(G\mathrm{app})$ we know that:

$$(G\mathrm{app})\frac{\Gamma; \Sigma \vdash t'_1 : G'_1 \qquad \Gamma; \Sigma \vdash t'_2 : G'_2 \qquad G'_2 \sim \widetilde{dom}(G'_1)}{\Gamma; \Sigma \vdash t'_1\, t'_2 : \widetilde{cod}(G'_1)} \tag{C.5}$$

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

$$(\mathrm{Papp})\frac{t'_1 \sqsubseteq t''_1 \qquad t'_2 \sqsubseteq t''_2}{t'_1\, t'_2 \sqsubseteq t''_1\, t''_2} \tag{C.6}$$

Using induction hypotheses on the premises of C.5, $\Gamma; \Sigma \vdash t''_1 : G''_1$ and $\Gamma; \Sigma \vdash t''_2 : G''_2$, where $G'_1 \sqsubseteq G''_1$ and $G'_2 \sqsubseteq G''_2$. By definition precision (Definition 2) and the definition of $\widetilde{dom}$, $\widetilde{dom}(G'_1) \sqsubseteq \widetilde{dom}(G''_1)$ and, therefore by Lemma 60, $G''_2 \sim \widetilde{dom}(G''_1)$. Also, by the previous argument $\widetilde{cod}(G'_1) \sqsubseteq \widetilde{cod}(G''_1)$ Then we can use rule $(G\mathrm{app})$ to derive:

$$(G\mathrm{app})\frac{\Gamma'; \Sigma' \vdash t''_1 : G''_1 \qquad \Gamma'; \Sigma' \vdash t''_2 : G''_2 \qquad G''_2 \sim \widetilde{dom}(G''_1)}{\Gamma'; \Sigma' \vdash t''_1\, t''_2 : \widetilde{cod}(G''_1)}$$

and the result holds.

Case *(Gif)*. Then $t_1 = \mathsf{if}\ t'_1\ \mathsf{then}\ t'_2\ \mathsf{else}\ t'_3$ and $G_1 = (G'_2 \sqcap G'_3)$. By $(G\mathrm{if})$ we know that:

$$(G\mathrm{if})\frac{\Gamma; \Sigma \vdash t'_1 : G'_1 \qquad \Gamma; \Sigma \vdash t'_2 : G'_2 \qquad \Gamma; \Sigma \vdash t'_3 : G'_3}{\Gamma; \Sigma \vdash \mathsf{if}\ t'_1\ \mathsf{then}\ t'_2\ \mathsf{else}\ t'_3 : (G'_2 \sqcap G'_3)} \tag{C.7}$$

Consider t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = \text{if } t''_1 \text{ then } t''_2 \text{ else } t''_3$ and therefore

$$\text{(Pif)}\frac{t'_1 \sqsubseteq t''_1 \qquad t'_1 \sqsubseteq t''_1 \qquad t'_2 \sqsubseteq t''_2}{\text{if } t'_1 \text{ then } t'_2 \text{ else } t'_3 \sqsubseteq \text{if } t''_1 \text{ then } t''_2 \text{ else } t''_3} \tag{C.8}$$

Then we can use induction hypotheses on the premises of C.7 and derive:

$$\text{(Gif)}\frac{\Gamma';\,;\,\Sigma' \vdash t''_1 : G''_1 \qquad \Gamma';\,;\,\Sigma' \vdash t''_2 : G''_2 \qquad \Gamma';\,;\,\Sigma' \vdash t''_3 : G''_3}{\Gamma';\,;\,\Sigma' \vdash \text{if } t''_1 \text{ then } t''_2 \text{ else } t''_3 : (G''_2 \sqcap G''_3)}$$

Where $G'_1 \sqsubseteq G''_1$ and $G'_2 \sqsubseteq G''_2$. Using the definition of type precision (Definition 2) we can infer that

$$(G'_1 \sqcap G'_2) \sqsubseteq (G''_1 \sqcap G''_2)$$

and the result holds.

Case *(G::)*. Then $t_1 = t :: G_1$. By $(G::)$ we know that:

$$(G::)\frac{\Gamma \vdash t : G'_1 \qquad G'_1 \sim G_1}{\Gamma \vdash t :: G'_1 : G_1} \tag{C.9}$$

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

$$\text{(P::)}\frac{t \sqsubseteq t' \qquad G_1 \sqsubseteq G_2}{t :: G_1 \sqsubseteq t' :: G_2} \tag{C.10}$$

Using induction hypotheses on the premises of C.9, $\Gamma \vdash t' : G'_2$ where $G'_1 \sqsubseteq G'_2$. We can use rule $(G::)$ and Lemma 60 to derive:

$$(G::)\frac{\Gamma \vdash t' : G'_2 \qquad G'_2 \sim G_2}{\Gamma \vdash t' :: G_2 : G_2}$$

Where $G_1 \sqsubseteq G_2$ and the result holds.

Case *(Gref)*. Then $t_1 = \text{ref}^G\ t'_1$. By $(G\text{ref})$ we know that:

$$(G\text{ref})\frac{\Gamma;\,\Sigma \vdash t'_1 : G'_1 \qquad G'_1 \sim G}{\Gamma;\,\Sigma \vdash \text{ref } t'_1 : \text{Ref } G} \tag{C.11}$$

Consider t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = \text{ref}^{G'}\ t'_2$ and therefore

$$\text{(Pref)}\frac{t'_1 \sqsubseteq t'_2 \qquad G \sqsubseteq G'}{\text{ref}^G\ t'_1 \sqsubseteq \text{ref}^{G'}\ t'_2} \tag{C.12}$$

Using induction hypotheses on the premise of C.11, $\Gamma;\Sigma \vdash t'_1 : G'_1$, where $G'_1 \sqsubseteq G'_2$. By definition of precision on types $\text{Ref } G \sqsubseteq \text{Ref } G'$. Then we can use rule $(G\text{ref})$ to derive:

$$(G\text{ref})\frac{\Gamma';\,\Sigma' \vdash t'_2 : G'_2 \qquad G'_2 \sim G'}{\Gamma';\,\Sigma' \vdash \text{ref } t'_2 : \text{Ref } G'}$$

and the result holds.

Case *(Gderef)*. Then $t_1 = !t'_1$. By $(G\text{deref})$ we know that:

$$(G\text{deref})\frac{\Gamma;\,\Sigma \vdash t'_1 : G}{\Gamma;\,\Sigma \vdash !t'_1 : \widetilde{tref}(G)} \tag{C.13}$$

Consider t_2 such that $t_1 \sqsubseteq t_2$. By definition of term precision t_2 must have the form $t_2 = !t'_2$ and therefore

$$\text{(Pref)}\frac{t'_1 \sqsubseteq t'_2}{!t'_1 \sqsubseteq !t'_2} \tag{C.14}$$

$$\frac{}{\Omega \cup \{x^{G_1} \sqsubseteq x^{G_2}\} \vdash x^{G_1} \sqsubseteq x^{G_2}} \qquad \frac{}{\Omega \vdash b \sqsubseteq b} \qquad (\sqsubseteq_{APP})\,\frac{\Omega \vdash t^{G_{11}} \sqsubseteq t^{G_{21}} \quad \Omega \vdash t^{G_{12}} \sqsubseteq t^{G_{22}} \quad \varepsilon_{11} \sqsubseteq \varepsilon_{21} \quad \varepsilon_{12} \sqsubseteq \varepsilon_{22} \quad G_1 \sqsubseteq G_2}{\Omega \vdash \varepsilon_{11}t^{G_{11}} \mathbin{@^{G_1}} \varepsilon_{12}t^{G_{12}} \sqsubseteq \varepsilon_{21}t^{G_{21}} \mathbin{@^{G_2}} \varepsilon_{22}t^{G_{22}}}$$

$$(\sqsubseteq_{\oplus})\,\frac{\Omega \vdash t^{G_{11}} \sqsubseteq t^{G_{21}} \quad \Omega \vdash t^{G_{12}} \sqsubseteq t^{G_{22}} \quad \varepsilon_{11} \sqsubseteq \varepsilon_{21} \quad \varepsilon_{12} \sqsubseteq \varepsilon_{22}}{\Omega \vdash (\varepsilon_{11}t^{G_{11}} \oplus \varepsilon_{12}t^{G_{12}}) \sqsubseteq (\varepsilon_{21}t^{G_{21}} \oplus \varepsilon_{22}t^{G_{22}})}$$

$$(\sqsubseteq_{IF})\,\frac{\Omega \vdash t^{G_{11}} \sqsubseteq t^{G_{21}} \quad \varepsilon_{11} \sqsubseteq \varepsilon_{21} \quad \Omega \vdash t^{G_{12}} \sqsubseteq t^{G_{23}} \quad \varepsilon_{12} \sqsubseteq \varepsilon_{22} \quad \Omega \vdash t^{G_{13}} \sqsubseteq t^{G_{23}} \quad \varepsilon_{13} \sqsubseteq \varepsilon_{23}}{\Omega \vdash \text{if } \varepsilon_{11}t^{G_{11}} \text{ then } \varepsilon_{12}t^{G_{12}} \text{ else } \varepsilon_{13}t^{G_{13}} \sqsubseteq \text{if } \varepsilon_{21}t^{G_{21}} \text{ then } \varepsilon_{22}t^{G_{22}} \text{ else } \varepsilon_{23}t^{G_{23}}}$$

$$(\sqsubseteq_{\lambda})\,\frac{G_{11} \sqsubseteq G_{12} \quad \Omega \cup \{x^{G_{11}} \sqsubseteq x^{G_{12}}\} \vdash t^{G_{12}} \sqsubseteq t^{G_{22}}}{\Omega \vdash (\lambda x^{G_{11}}.t^{G_{12}}) \sqsubseteq (\lambda x^{G_{21}}.t^{G_{22}})} \qquad (\sqsubseteq_{::})\,\frac{\Omega \vdash t^{G_{11}} \sqsubseteq t^{G_{21}} \quad G_{12} \sqsubseteq G_{22} \quad \varepsilon_1 \sqsubseteq \varepsilon_2}{(\varepsilon_1 t^{G_{11}} :: G_{12}) \sqsubseteq (\varepsilon_2 t^{G_{21}} :: G_{22})}$$

$$(\sqsubseteq_{REF})\,\frac{\Omega \vdash t^{G_{12}} \sqsubseteq t^{G_{22}} \quad \varepsilon_1 \sqsubseteq \varepsilon_2 \quad G_{11} \sqsubseteq G_{21}}{\text{ref}^{G_{11}}\ \varepsilon_1 t^{G_{12}} \sqsubseteq \text{ref}^{G_{21}}\ \varepsilon_2 t^{G_{22}}} \qquad (\sqsubseteq_{!})\,\frac{\Omega \vdash t^{G_{12}} \sqsubseteq t^{G_{22}} \quad \varepsilon_1 \sqsubseteq \varepsilon_2 \quad G_{11} \sqsubseteq G_{21}}{!^{G_{11}} \varepsilon_1 t^{G_{12}} \sqsubseteq !^{G_{21}} \varepsilon_2 t^{G_{22}}}$$

$$(\sqsubseteq_{:=})\,\frac{\Omega \vdash t^{G_{11}} \sqsubseteq t^{G_{21}} \quad \Omega \vdash t^{G_{12}} \sqsubseteq t^{G_{22}} \quad \varepsilon_{11} \sqsubseteq \varepsilon_{21} \quad \varepsilon_{12} \sqsubseteq \varepsilon_{22} \quad G_1 \sqsubseteq G_2}{\Omega \vdash \varepsilon_{11}t^{G_{11}} :=^{G_1} \varepsilon_{12}t^{G_{12}} \sqsubseteq \varepsilon_{21}t^{G_{21}} :=^{G_2} \varepsilon_{22}t^{G_{22}}} \qquad (\sqsubseteq_{o})\,\frac{}{\Omega \cup \{o^{G_1} \sqsubseteq o^{G_2}\} \vdash o^{G_1} \sqsubseteq o^{G_2}}$$

$$(\sqsubseteq_{\mu})\,\frac{\forall o^{G_1} \in dom(\mu_1).\exists o^{G_2} \in dom(\mu_2)\ s.t. \quad \Omega \vdash o^{G_1} \sqsubseteq o^{G_2} \quad \Omega \vdash \mu_1(o^{G_1}) \sqsubseteq \mu_2(o^{G_2})}{\Omega \vdash \mu_1 \sqsubseteq \mu_2}$$

Fig. C.18. Intrinsic term precision.

Using induction hypotheses on the premise of C.13, $\Gamma; \Sigma \vdash t'_1 : G'_1$. By definition of $\widetilde{tref}$, $\widetilde{tref}\, G \sqsubseteq \widetilde{tref}\, G'$. Then we can use rule (Gderef) to derive:

$$(G\text{deref})\,\frac{\Gamma'; \Sigma' \vdash t'_2 : G'}{\Gamma'; \Sigma' \vdash !t'_2 : \widetilde{tref}(G')}$$

and the result holds.

Case *(Gassign).* Then $t_1 = t'_1 := t'_2$ and $G_1 = G_{12}$. By (Gassign) we know that:

$$(G\text{assign})\,\frac{\Gamma; \Sigma \vdash t'_1 : G'_1 \quad \Gamma; \Sigma \vdash t'_2 : G'_2 \quad G'_2 \sim \widetilde{tref}(G'_1)}{\Gamma; \Sigma \vdash t'_1 := t'_2 : \widetilde{cod}(G'_1)} \tag{C.15}$$

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

$$(\text{Passign})\,\frac{t'_1 \sqsubseteq t''_1 \quad t'_2 \sqsubseteq t''_2}{t'_1 := t'_2 \sqsubseteq t''_1 := t''_2} \tag{C.16}$$

Using induction hypotheses on the premises of C.15, $\Gamma; \Sigma \vdash t''_1 : G''_1$ and $\Gamma; \Sigma \vdash t''_2 : G''_2$, where $G'_1 \sqsubseteq G''_1$ and $G'_2 \sqsubseteq G''_2$. By definition precision (Definition 2) and the definition of $\widetilde{tref}$, $\widetilde{tref}(G'_1) \sqsubseteq \widetilde{tref}(G''_1)$ and, therefore by Lemma 60, $G''_2 \sim \widetilde{tref}(G''_1)$. Then we can use rule (Gassign) to derive:

$$(G\text{assign})\,\frac{\Gamma'; \Sigma' \vdash t''_1 : G''_1 \quad \Gamma'; \Sigma' \vdash t''_2 : G''_2 \quad G''_2 \sim \widetilde{tref}(G''_1)}{\Gamma'; \Sigma' \vdash t''_1 := t''_2 : \text{Unit}}$$

and the result holds. □

C.3. Dynamic gradual guarantee

In this section we present the proof the Dynamic Gradual Guarantee for $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$.

Definition 14 *(Intrinsic term precision).* Let $\Omega \in \mathscr{P}(\mathbb{V}[*] \times \mathbb{V}[*]) \cup \mathscr{P}(\text{Loc}_* \times \text{Loc}_*)$ be defined as $\Omega ::= \{\overline{x^{G_{i1}} \sqsubseteq x^{G_{i2}}, o^{G_{j1}} \sqsubseteq o^{G_{j2}}}\}$ We define an ordering relation $(\cdot \vdash \cdot \sqsubseteq \cdot) \in (\mathscr{P}(\mathbb{V}[*] \times \mathbb{V}[*]) \cup \mathscr{P}(\text{Loc}_* \times \text{Loc}_*)) \times \mathbb{T}[*] \times \mathbb{T}[*]$ shown in Fig. C.18.

Definition 15 *(Well Formedness of* Ω*).* We say that Ω is well formed iff $\forall \{x^{G_{i1}} \sqsubseteq x^{G_{i2}}\} \in \Omega. G_{i1} \sqsubseteq G_{i2}$.

Before proving the gradual guarantee, we first establish some auxiliary properties of precision. For the following propositions, we assume Well Formedness of Ω (Definition 15).

Proposition 62. *If $\Omega \vdash t^{G_1} \sqsubseteq t^{G_2}$ for some $\Omega \in \mathscr{P}(\mathbb{V}[*] \times \mathbb{V}[*]) \cup \mathscr{P}(\mathrm{Loc}_* \times \mathrm{Loc}_*)$, then $G_1 \sqsubseteq G_2$.*

Proof. Straightforward induction on $\Omega \vdash t^{G_1} \sqsubseteq t^{G_2}$, since the corresponding precision on types is systematically a premise (either directly or transitively). $\square$

Proposition 63 *(Substitution preserves precision). If $\Omega \cup \{x^{G_3} \sqsubseteq x^{G_4}\} \vdash t^{G_1} \sqsubseteq t^{G_2}$ and $\Omega \vdash t^{G_3} \sqsubseteq t^{G_4}$, then $\Omega \vdash [t^{G_3}/x^{G_3}]t^{G_1} \sqsubseteq [t^{G_4}/x^{G_4}]t^{G_2}$.*

Proof. By induction on the derivation of $t^{G_1} \sqsubseteq t^{G_2}$, and case analysis of the last rule used in the derivation. All cases follow either trivially (no premises) or by the induction hypotheses. $\square$

Proposition 64 *(Monotonicity of evidence). If $\varepsilon_1 \sqsubseteq \varepsilon_2$, $\varepsilon_3 \sqsubseteq \varepsilon_4$, and $\varepsilon_1 \circ^{=} \varepsilon_3$ is defined, then $\varepsilon_1 \circ^{=} \varepsilon_3 \sqsubseteq \varepsilon_2 \circ^{=} \varepsilon_4$.*

Proof. By definition of consistent transitivity for $=$ and the definition of precision. $\square$

Proposition 65. *If $G_{11} \sqsubseteq G_{12}$ and $G_{21} \sqsubseteq G_{22}$ then $G_{11} \sqcap G_{21} \sqsubseteq G_{12} \sqcap G_{22}$.*

Proof. By induction on the type derivation of the types and meet. $\square$

Proposition 66 *(Dynamic guarantee for $\longrightarrow$). Suppose $\Omega \vdash t_1^{G_1} \sqsubseteq t_1^{G_2}$ and $\mu_1 \sqsubseteq \mu_2$. If $t_1^{G_1} \mid \mu_1 \longrightarrow t_2^{G_1} \mid \mu_1'$ then $t_1^{G_2} \mid \mu_2 \longrightarrow t_2^{G_2} \mid \mu_2'$, where $\Omega' \vdash t_2^{G_1} \sqsubseteq t_2^{G_2}$, $\mu_1' \sqsubseteq \mu_2'$ for some $\Omega' \supseteq \Omega$.*

Proof. By induction on reduction $t_1^{G_1} \mid \mu_1 \longrightarrow t_2^{G_1} \mid \mu_1'$. For simplicity we omit the $\Omega \vdash$ notation on precision relations when it is not relevant for the argument.

Case *(r1)*. We know that $t_1^{G_1} = (\varepsilon_{11}(c_1) \oplus \varepsilon_{12}(c_2))$ then by $(\sqsubseteq_{\oplus})$ $t_1^{G_2} = (\varepsilon_{21}(c_1) \oplus \varepsilon_{22}(c_2))$ for some $\varepsilon_{21}, \varepsilon_{22}$ such that $\varepsilon_{11} \sqsubseteq \varepsilon_{21}$ and $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$.

If $t_1^{G_1} \mid \mu_1 \longrightarrow c_3 \mid \mu_1$ where $c_3 = (c_1 \llbracket\oplus\rrbracket c_2)$, then $t_1^{G_2} \mid \mu_2 \longrightarrow c_3' \mid \mu_2$ where $c_3' = (c_1 \llbracket\oplus\rrbracket c_2)$. But $c_3 = c_3'$ and therefore $t_2^{G_1} \sqsubseteq t_2^{G_2}$ and the result holds.

Case *(r2)*. We know that $t_1^{G_1} = \varepsilon_{11}(\lambda x^{G_{11}}.t^{G_{12}}) \,@^{G_1 \longrightarrow G_2}\, \varepsilon_{12} u$ then by $(\sqsubseteq_{app})$ $t_1^{G_2}$ must have the form $t_1^{G_2} = \varepsilon_{21}(\lambda x^{G_{21}}.t^{G_{22}}) \,@^{G_3 \longrightarrow G_4}\, \varepsilon_{22} u_2$ for some $\varepsilon_{21}, x^{G_{21}}, t^{G_{22}}, G_3, G_4, \varepsilon_{22}$ and u_2.

Let us pose $\varepsilon_1 = \varepsilon_{12} \circ^{=} \mathit{idom}(\varepsilon_{11})$. Then

$$t_1^{G_1} \mid \mu_1 \longrightarrow \mathit{icod}(\varepsilon_{11}) t_1' :: G_2 \mid \mu_1 \text{ with } t_1' = [(\varepsilon_1 u_1 :: G_{11})/x^{G_{11}}]t^{G_{12}}.$$

Also, by 64, $\varepsilon_2 = \varepsilon_{22} \circ^{=} \mathit{idom}(\varepsilon_{21})$ is defined. Then

$$t_1^{G_2} \mid \mu_2 \longrightarrow \mathit{icod}(\varepsilon_{21}) t_2' :: G_4 \mid \mu_2 \text{ with } t_2' = [(\varepsilon_2 u_2 :: G_{21})/x^{G_{21}}]t^{G_{22}}.$$

As $\Omega \vdash t_1^{G_1} \sqsubseteq t_1^{G_2}$, then $u_1 \sqsubseteq u_2$, $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$ and $\mathit{idom}(\varepsilon_{11}) \sqsubseteq \mathit{idom}(\varepsilon_{21})$ as well, then by Proposition 64 $\varepsilon_1 \sqsubseteq \varepsilon_2$. Then $\varepsilon_1 u_1 :: G_{11} \sqsubseteq \varepsilon_2 u_2 :: G_{21}$ by $(\sqsubseteq_{::})$.

We also know by $(\sqsubseteq_{APP})$ and $(\sqsubseteq_{\lambda})$ that $\Omega \cup \{x^{G_{21}} \sqsubseteq x^{G_{21}}\} \vdash t^{G_{12}} \sqsubseteq t^{G_{22}}$. By Substitution preserves precision (Proposition 63) $t_1' \sqsubseteq t_2'$, therefore $\mathit{icod}(\varepsilon_{11}) t_1' :: G_2 \sqsubseteq \mathit{icod}(\varepsilon_{21}) t_2' :: G_4$ by $(\sqsubseteq_{::})$. Then $t_2^{G_1} \sqsubseteq t_2^{G_2}$.

Case *(r3 − true)*. $t_1^{G_1} =$ if ε_{11}true then $\varepsilon_{12} t^{G_{12}}$ else $\varepsilon_{13} t^{G_{13}}$ then by $(\sqsubseteq_{if})$ $t_1^{G_2}$ has the form $t_1^{G_2} =$ if ε_{21} true then $\varepsilon_{22} t^{G_{22}}$ else $\varepsilon_{23} t^{G_{23}}$ for some $\varepsilon_{21}, \varepsilon_{22}, t^{G_22}, \varepsilon_{23}$, and $t^{G_{32}}$. Then $t_1^{G_1} \mid \mu_1 \longrightarrow \varepsilon_{12} t^{G_{12}} :: (G_{12} \sqcap G_{13}) \mid \mu_1$, and $t_1^{G_2} \mid \mu_2 \longrightarrow \varepsilon_{22} t^{G_{22}} :: (G_{22} \sqcap G_{23}) \mid \mu_2$. Using the fact that $t_1^{G_1} \sqsubseteq t_2^{G_2}$ we know that $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$, $t^{G_{12}} \sqsubseteq t^{G_{22}}$ and by Proposition 62, $G_{12} \sqsubseteq G_{22}$ and $G_{13} \sqsubseteq G_{23}$. Therefore by Proposition 65 $(G_{12} \sqcap G_{13}) \sqsubseteq (G_{22} \sqcap G_{23})$. Then using $(\sqsubseteq_{::})$, $t_2^{G_1} \sqsubseteq t_2^{G_2}$.

Case *(r3 − false)*. Same as case $\longrightarrow$if-true, using the fact that $\varepsilon_{13} \sqsubseteq \varepsilon_{23}$ and $t^{G_{13}} \sqsubseteq t^{G_{23}}$.

Case *(r4)*. We know that $t_1^{G_1} = \mathsf{ref}^{G_1'}\ \varepsilon_1 u_1$ where $G_1 = \mathsf{Ref}\ G_1'$, then by $(\sqsubseteq_{REF})$ $t_1^{G_2}$ must have the form $t_1^{G_2} = \mathsf{ref}^{G_2'}\ \varepsilon_2 u_2$ for some ε_2, u_2, G_2' such that $\varepsilon_1 \sqsubseteq \varepsilon_2$, $u_1 \sqsubseteq u_2$, and $G_1' \sqsubseteq G_2'$.

Then

$$t_1^{G_1} \mid \mu_1 \longrightarrow o^{G_1'} \mid \mu_1[o^{G_1'} \mapsto \varepsilon_1 u_1 :: G_1'].$$

Also, $t_1^{G_2} \mid \mu_2 \longrightarrow o^{G_2'} \mid \mu_2[o^{G_2'} \mapsto \varepsilon_2 u_2 :: G_2']$.

Then by $(\sqsubseteq_{::})$, $\varepsilon_1 u_1 :: G_1' \sqsubseteq \varepsilon_2 u_2 :: G_2'$, and then $\mu_1[o^{G_1'} \mapsto \varepsilon_1 u_1 :: G_1'] \sqsubseteq \mu_2[o^{G_2'} \mapsto \varepsilon_2 u_2 :: G_2']$. Also by $(\sqsubseteq_o)$, as $G_1' \sqsubseteq G_2'$, $o^{G_1'} \sqsubseteq o^{G_2'}$ and the result holds.

Case *(r5).* We know that $t_1^{G_1} = !^{G_1}\varepsilon_1 o^{G_1'}$, then by $(\sqsubseteq_!)$ $t_1^{G_2}$ must have the form $t_1^{G_2} = !^{G_2}\varepsilon_2 o^{G_2'}$ for some $\varepsilon_2, o^{G_2'}, G_2'$ such that $\varepsilon_1 \sqsubseteq \varepsilon_2, o^{G_1'} \sqsubseteq o^{G_2'}$, and $G_1' \sqsubseteq G_2'$.

Then $t_1^{G_1} \mid \mu_1 \longrightarrow \varepsilon_1\mu_1(o^{G_1'}) :: G_1 \mid \mu_1$.

Also, $t_1^{G_2} \mid \mu_2 \longrightarrow \varepsilon_2\mu_2(o^{G_2'}) :: G_2 \mid \mu_2$.

As $\mu_1 \sqsubseteq \mu_2$, then $\mu_1(o^{G_1'}) \sqsubseteq \mu_2(o^{G_2'})$. Then by $(\sqsubseteq_{::})$, $\varepsilon_1\mu_1(o^{G_1'}) :: G_1' \sqsubseteq \varepsilon_2\mu_2(o^{G_2'}) :: G_2'$, and the result holds.

Case *(r6).* We know that $t_1^{G_1} = \varepsilon_{11} o^{G_{11}} :=^{G_{12}} \varepsilon_{12} u_1$ where $G_1 = \mathsf{Unit}$, then by $(\sqsubseteq_{:=})$ $t_1^{G_2}$ must have the form $t_1^{G_2} = \varepsilon_{21} o^{G_{21}} :=^{G_{22}} \varepsilon_{22} u_2$ for some $\varepsilon_{21}, \varepsilon_{22}, u_2, G_{21}, G_{22}$ such that $\varepsilon_{11} \sqsubseteq \varepsilon_{21}, \varepsilon_{12} \sqsubseteq \varepsilon_{22}, u_1 \sqsubseteq u_2, G_{11} \sqsubseteq G_{21}, G_{12} \sqsubseteq G_{22}$.

Let us pose $\varepsilon_1 = \varepsilon_{12} \circ^{=} \mathit{iref}(\varepsilon_{11})$. Then $t_1^{G_1} \mid \mu_1 \longrightarrow \mathsf{unit} \mid \mu_1[o^{G_{11}} \mapsto \varepsilon_1 u_1 :: G_{11}]$.

By inspection of evidence and inversion lemma, as $\varepsilon_{12} \sqsubseteq \varepsilon_{21}$ then $\mathit{iref}(\varepsilon_{12}) \sqsubseteq \mathit{iref}(\varepsilon_{21})$. Also, by 64, $\varepsilon_2 = \varepsilon_{22} \circ^{=} \mathit{iref}(\varepsilon_{21})$ is defined and $\varepsilon_1 \sqsubseteq \varepsilon_2$. Then, $t_1^{G_2} \mid \mu_2 \longrightarrow \mathsf{unit} \mid \mu_2[o^{G_{21}} \mapsto \varepsilon_2 u_2 :: G_{21}]$.

Then by $(\sqsubseteq_{::})$, $\varepsilon_1 u_1 :: G_{11} \sqsubseteq \varepsilon_2 u_2 :: G_{21}$, and then $\mu_1[o^{G_{21}} \mapsto \varepsilon_1 u_1 :: G_{11}] \sqsubseteq \mu_2[o^{G_{21}} \mapsto \varepsilon_2 u_2 :: G_{21}]$ and the result holds. $\square$

Proposition 67 *(Dynamic gradual guarantee). Suppose $t_1^{G_1} \sqsubseteq t_1^{G_2}$ and $\mu_1 \sqsubseteq \mu_2$. Then if $t_1^{G_1} \mid \mu_1 \longmapsto t_2^{G_1} \mid \mu_1'$ then $t_1^{G_2} \mid \mu_2 \longmapsto t_2^{G_2} \mid \mu_2'$ where $t_2^{G_1} \sqsubseteq t_2^{G_2}$ and $\mu_1' \sqsubseteq \mu_2'$.*

Proof. We prove the following property instead: Suppose $\Omega \vdash t_1^{G_1} \sqsubseteq t_1^{G_2}$ and $\mu_1 \sqsubseteq \mu_2$. If $t_1^{G_1} \mid \mu_1 \longmapsto t_2^{G_1} \mid \mu_1'$ then $t_1^{G_2} \mid \mu_2 \longmapsto t_2^{G_2} \mid \mu_2'$ where $\Omega' \vdash t_2^{G_1} \sqsubseteq t_2^{G_2}$, and $\mu_1' \sqsubseteq \mu_2'$ for some $\Omega' \supseteq \Omega$.

By induction on reduction $t_1^{G_1} \mid \mu_1 \longmapsto t_2^{G_1} \mid \mu_1'$. For simplicity we omit the $\Omega \vdash$ notation on precision relations when it is not relevant for the argument.

Case $(t_1^{G_1} \mid \mu_1 \longrightarrow t_2^{G_1} \mid \mu_1')$. By dynamic guarantee of $\longrightarrow$ (Proposition 66), $t_1^{G_2} \mid \mu_2 \longrightarrow t_2^{G_2} \mid \mu_2'$ where $\Omega' \vdash t_2^{G_1} \sqsubseteq t_2^{G_2}$, $\mu_1' \sqsubseteq \mu_2'$ for some $\Omega' \supseteq \Omega$. And the result holds immediately.

Case $(\varepsilon_{11} t_{11}^{G_{11}} @^{G_{13}} \varepsilon_{12} t_{12}^{G_{12}} \mid \mu_1 \longmapsto \varepsilon_{11}' t_{11}'^{G_{11}} @^{G_{13}} \varepsilon_{12} t_{12}^{G_{12}} \mid \mu_1')$. By inspection of $(\sqsubseteq_{APP})$ $t^{G_2} = \varepsilon_{21} t_{21}^{G_{21}} @^{G_{23}} \varepsilon_{22} t_{22}^{G_{22}}$, where $\varepsilon_{11} \sqsubseteq \varepsilon_{21}$, $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$ $G_{13} \sqsubseteq G_{23}$, $t_{11}^{G_{11}} \sqsubseteq t_{21}^{G_{21}}$, and $t_{12}^{G_{12}} \sqsubseteq t_{22}^{G_{22}}$. By induction hypothesis on $\varepsilon_{11} t_{11}^{G_{11}} :: G_{13} \mid \mu_1 \longmapsto \varepsilon_{11}' t_{11}'^{G_{11}} :: G_{13} \mid \mu_1'$, then $\varepsilon_{21} t_{21}^{G_{21}} :: G_{23} \mid \mu_2 \longmapsto \varepsilon_{21}' t_{21}'^{G_{21}} :: G_{23} \mid \mu_2'$, where $t_{11}'^{G_{11}} \sqsubseteq t_{21}'^{G_{21}}$, $\varepsilon_{11}' \sqsubseteq \varepsilon_{21}'$ and $\mu_1' \sqsubseteq \mu_2'$. Then by $(\sqsubseteq_{APP})$ and Lemma 92, $\varepsilon_{11}' t_{11}'^{G_{11}} @^{G_{13}} \varepsilon_{12} t_{12}^{G_{12}} \sqsubseteq \varepsilon_{21}' t_{21}'^{G_{21}} @^{G_{23}} \varepsilon_{22} t_{22}^{G_{22}}$ and the result holds.

Case $(\varepsilon_{11} u_1 @^{G_{13}} \varepsilon_{12} t_{12}^{G_{12}} \mid \mu_1 \longmapsto \varepsilon_{11} u_1 @^{G_{13}} \varepsilon_{12}' t_{12}'^{G_{12}} \mid \mu_1')$. By inspection of $(\sqsubseteq_{APP})$ $t^{G_2} = \varepsilon_{21} u_2 @^{G_{23}} \varepsilon_{22} t_{22}^{G_{22}}$, where $\varepsilon_{11} \sqsubseteq \varepsilon_{21}$, $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$ $G_{13} \sqsubseteq G_{23}$, $u_1 \sqsubseteq u_2$, and $t_{12}^{G_{12}} \sqsubseteq t_{22}^{G_{22}}$. By induction hypothesis on $\varepsilon_{12} t_{12}^{G_{12}} :: \widetilde{dom}(G_{13}) \mid \mu_1 \longmapsto \varepsilon_{12}' t_{12}'^{G_{12}} :: \widetilde{dom}(G_{13}) \mid \mu_1'$, then $\varepsilon_{22} t_{22}^{G_{22}} :: \widetilde{dom}(G_{23}) \mid \mu_2 \longmapsto \varepsilon_{22}' t_{22}'^{G_{22}} :: \widetilde{dom}(G_{23}) \mid \mu_2$, where $t_{12}'^{G_{12}} \sqsubseteq t_{22}'^{G_{22}}$, $\varepsilon_{12}' \sqsubseteq \varepsilon_{22}'$, and $\mu_1' \sqsubseteq \mu_2'$. Then by $(\sqsubseteq_{APP})$ and Lemma 93, $\varepsilon_{11} u_1 @^{G_{13}} \varepsilon_{12} t_{12}^{G_{12}} \sqsubseteq \varepsilon_{21} u_2 @^{G_{23}} \varepsilon_{22} t_{22}^{G_{22}}$ and the result holds.

Case $(\mathsf{ref}^{G_{12}}\ \varepsilon_1 t^{G_{11}} \mid \mu_1 \longmapsto \mathsf{ref}^{G_{12}}\ \varepsilon_1' t'^{G_{11}} \mid \mu_1')$. By inspection of $(\sqsubseteq_{REF})$ $t^{G_2} = \mathsf{ref}^{G_{22}}\ \varepsilon_2 t^{G_{21}}$, where $\varepsilon_1 \sqsubseteq \varepsilon_2$, $G_{11} \sqsubseteq G_{21}$, $G_{12} \sqsubseteq G_{22}$, $t^{G_{11}} \sqsubseteq t^{G_{21}}$. By induction hypothesis on $\varepsilon_1 t^{G_{11}} :: G_{12} \mid \mu_1 \longmapsto \varepsilon_1' t'^{G_{11}} :: G_{12} \mid \mu_1'$, then $\varepsilon_2 t^{G_{21}} :: G_{22} \mid \mu_2 \longmapsto \varepsilon_2' t'^{G_{21}} :: G_{22} \mid \mu_2'$, where $t'^{G_{11}} \sqsubseteq t'^{G_{21}}$, $\varepsilon_1' \sqsubseteq \varepsilon_2'$, and $\mu_1' \sqsubseteq \mu_2'$. Then by $(\sqsubseteq_{REF})$ and Lemma 96, $\mathsf{ref}^{G_{12}}\ \varepsilon_1' t'^{G_{11}} \sqsubseteq \mathsf{ref}^{G_{22}}\ \varepsilon_2' t'^{G_{21}}$ and the result holds.

Case $(!^{G_{12}}\varepsilon_1 t^{G_{11}} \mid \mu_1 \longmapsto !^{G_{12}}\varepsilon_1' t'^{G_{11}} \mid \mu_1')$. By inspection of $(\sqsubseteq_!)$ $t^{G_2} = !^{G_{22}}\varepsilon_2 t^{G_{21}}$, where $\varepsilon_1 \sqsubseteq \varepsilon_2$, $G_{11} \sqsubseteq G_{21}$, $G_{12} \sqsubseteq G_{22}$, $t^{G_{11}} \sqsubseteq t^{G_{21}}$. By induction hypothesis on $\varepsilon_1 t^{G_{11}} :: G_{12} \mid \mu_1 \longmapsto \varepsilon_1' t'^{G_{11}} :: G_{12} \mid \mu_1'$, then $\varepsilon_2 t^{G_{21}} :: G_{22} \mid \mu_2 \longmapsto \varepsilon_2' t'^{G_{21}} :: G_{22} \mid \mu_2'$, where $t'^{G_{11}} \sqsubseteq t'^{G_{21}}$, $\varepsilon_1' \sqsubseteq \varepsilon_2'$, and $\mu_1' \sqsubseteq \mu_2'$. Then by $(\sqsubseteq_!)$ and Lemma 96, $!^{G_{12}}\varepsilon_1' t'^{G_{11}} \sqsubseteq !^{G_{22}}\varepsilon_2' t'^{G_{21}}$ and the result holds.

Case $(\varepsilon_{11} t_{11}^{G_{11}} :=^{G_{13}} \varepsilon_{12} t_{12}^{G_{12}} \mid \mu_1 \longmapsto \varepsilon_{11}' t_{11}'^{G_{11}} :=^{G_{13}} \varepsilon_{12} t_{12}^{G_{12}} \mid \mu_1')$. By inspection of $(\sqsubseteq_{:=})$ $t^{G_2} = \varepsilon_{21} t_{21}^{G_{21}} :=^{G_{23}} \varepsilon_{22} t_{22}^{G_{22}}$, where $\varepsilon_{11} \sqsubseteq \varepsilon_{21}$, $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$ $G_{13} \sqsubseteq G_{23}$, $t_{11}^{G_{11}} \sqsubseteq t_{21}^{G_{21}}$, and $t_{12}^{G_{12}} \sqsubseteq t_{22}^{G_{22}}$. By induction hypothesis on $\varepsilon_{11} t_{11}^{G_{11}} :: \mathsf{Ref}\ G_{13} \mid \mu_1 \longmapsto \varepsilon_{11}' t_{11}'^{G_{11}} :: \mathsf{Ref}\ G_{13} \mid \mu_1'$, then $\varepsilon_{21} t_{21}^{G_{21}} :: \mathsf{Ref}\ G_{23} \mid \mu_2 \longmapsto \varepsilon_{21}' t_{21}'^{G_{21}} :: \mathsf{Ref}\ G_{23} \mid \mu_2'$, where $t_{11}'^{G_{11}} \sqsubseteq t_{21}'^{G_{21}}$, $\varepsilon_{11}' \sqsubseteq \varepsilon_{21}'$ and $\mu_1' \sqsubseteq \mu_2'$. Then by $(\sqsubseteq_{:=})$ and Lemma 94, $\varepsilon_{11}' t_{11}'^{G_{11}} :=^{[}\ \varepsilon_{12} t_{12}^{G_{12}} G_{13}] \sqsubseteq \varepsilon_{21}' t_{21}'^{G_{21}} :=^{[}\ \varepsilon_{22} t_{22}^{G_{22}} G_{23}]$ and the result holds.

Case $(\varepsilon_{11} u_1 :=^{G_{13}} \varepsilon_{12} t_{12}^{G_{12}} \mid \mu_1 \longmapsto \varepsilon_{11} u_1 :=^{G_{13}} \varepsilon'_{12} t'^{G_{12}}_{12} \mid \mu'_1)$. By inspection of $(\sqsubseteq_{:=})$ $t^{G_2} = \varepsilon_{21} u_2 :=^{G_{23}} \varepsilon_{22} t_{22}^{G_{22}}$, where $\varepsilon_{11} \sqsubseteq \varepsilon_{21}$, $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$ $G_{13} \sqsubseteq G_{23}$, $u_1 \sqsubseteq u_2$, and $t_{12}^{G_{12}} \sqsubseteq t_{22}^{G_{22}}$. By induction hypothesis on $\varepsilon_{12} t_{12}^{G_{12}} :: G_{13} \mid \mu_1 \longmapsto \varepsilon'_{12} t'^{G_{12}}_{12} :: G_{13} \mid \mu'_1$, then $\varepsilon_{22} t_{22}^{G_{22}} :: G_{23} \mid \mu_2 \longmapsto \varepsilon'_{22} t'^{G_{22}}_{22} :: G_{23} \mid \mu_2$, where $t'^{G_{12}}_{12} \sqsubseteq t'^{G_{22}}_{22}$, $\varepsilon'_{12} \sqsubseteq \varepsilon'_{22}$, and $\mu'_1 \sqsubseteq \mu'_2$. Then by $(\sqsubseteq_{:=})$ and Lemma 95, $\varepsilon_{11} u_1 :=^{G_{13}} \varepsilon_{12} t_{12}^{G_{12}} \sqsubseteq \varepsilon_{21} u_2 :=^{G_{23}} \varepsilon_{22} t_{22}^{G_{22}}$ and the result holds.

Case $(\mathsf{if}\ \varepsilon_{11} t^{G_{11}}\ \mathsf{then}\ \varepsilon_{12} t^{G_{12}}\ \mathsf{else}\ \varepsilon_{13} t^{G_{13}} \mid \mu_1 \longmapsto \mathsf{if}\ \varepsilon'_{11} t'^{G_{11}}\ \mathsf{then}\ \varepsilon_{12} t^{G_{12}}\ \mathsf{else}\ \varepsilon_{13} t^{G_{13}} \mid \mu'_1)$. By inspection of $(\sqsubseteq_{IF})$ $t^{G_2} = \mathsf{if}\ \varepsilon_{21} t^{G_{21}}\ \mathsf{then}\ \varepsilon_{22} t^{G_{22}}\ \mathsf{else}\ \varepsilon_{23} t^{G_{23}}$, where $\varepsilon_{11} \sqsubseteq \varepsilon_{21}$, $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$, $\varepsilon_{13} \sqsubseteq \varepsilon_{23}$, $t^{G_{11}} \sqsubseteq t^{G_{21}}$, $t^{G_{12}} \sqsubseteq t^{G_{22}}$, $t^{G_{13}} \sqsubseteq t^{G_{23}}$. By induction hypothesis on $\varepsilon_{11} t^{G_{12}} :: \mathsf{Bool} \mid \mu_1 \longmapsto \varepsilon'_{11} t'^{G_{12}} :: \mathsf{Bool} \mid \mu'_1$, then $\varepsilon_{21} t^{G_{22}} :: \mathsf{Bool} \mid \mu_2 \longmapsto \varepsilon'_{21} t'^{G_{22}} :: \mathsf{Bool} \mid \mu'_2$, where $t'^{G_{11}} \sqsubseteq t'^{G_{21}}$, $\varepsilon'_{11} \sqsubseteq \varepsilon'_{21}$ and $\mu'_1 \sqsubseteq \mu'_2$. Then by $(\sqsubseteq_{IF})$ and Lemma 98, $\mathsf{if}\ \varepsilon'_{11} t'^{G_{11}}\ \mathsf{then}\ \varepsilon_{12} t^{G_{12}}\ \mathsf{else}\ \varepsilon_{13} t^{G_{13}} \sqsubseteq \mathsf{if}\ \varepsilon'_{21} t'^{G_{21}}\ \mathsf{then}\ \varepsilon_{22} t^{G_{22}}\ \mathsf{else}\ \varepsilon_{23} t^{G_{23}}$ and the result holds.

Case $(\varepsilon_1 t^{G_{11}} :: G_{12} \mid \mu_1 \longmapsto \varepsilon_1 t'^{G_{11}} :: G_{12} \mid \mu'_1)$. By inspection of $(\sqsubseteq_{::})$ $t^{G_2} = \varepsilon_2 t^{G_{21}} :: G_{22}$, where $\varepsilon_1 \sqsubseteq \varepsilon_2$, $G_{11} \sqsubseteq G_{21}$, $G_{12} \sqsubseteq G_{22}$, $t^{G_{11}} \sqsubseteq t^{G_{21}}$. By induction hypothesis on $t^{G_{12}} \mid \mu_1 \longmapsto t'^{G_{12}} \mid \mu'_1$, then $t^{G_{22}} \mid \mu_2 \longmapsto t'^{G_{12}} \mid \mu_2$, where $t'^{G_{12}} \sqsubseteq t'^{G_{22}}$, and $\mu'_1 \sqsubseteq \mu'_2$. Then by $(\sqsubseteq_{::})$, $\varepsilon_1 t'^{G_{11}} :: G_{12} \sqsubseteq \varepsilon_2 t'^{G_{21}} :: G_{22}$ and the result holds.

Case $(\varepsilon_{12}(\varepsilon_{11} u_1 :: G_{11}) :: G_{12} \mid \mu_1 \longmapsto \varepsilon'_{11} u_1 :: G_{12} \mid \mu_1)$. By inspection of $(\sqsubseteq_{::})$ $t^{G_2} = \varepsilon_{22}(\varepsilon_{21} u_2 :: G_{21}) :: G_{22}$, where $\varepsilon_{11} \sqsubseteq \varepsilon_{21}$, $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$, $G_{11} \sqsubseteq G_{21}$, $G_{12} \sqsubseteq G_{22}$, $u_1 \sqsubseteq u_2$. If $\varepsilon'_{11} = \varepsilon_{11} \circ \varepsilon_{12}$ is defined, then by Proposition 64 $\varepsilon'_{21} = \varepsilon_{21} \circ \varepsilon_{22}$ is also defined, and furthermore $\varepsilon'_{11} \sqsubseteq \varepsilon'_{21}$. Then $\varepsilon_{22}(\varepsilon_{21} u_2 :: G_{21}) :: G_{22} \mid \mu_2 \longmapsto \varepsilon'_{21} u_2 :: G_{22} \mid \mu_2$, and the result holds directly by $(\sqsubseteq_{REF})$. □

Appendix D. Space efficiency

Lemma 68. $\forall \varepsilon \vdash G_1 \sim G_2, size(\varepsilon) \leq 2^{height(\varepsilon)} - 1$

Proof. By induction on ε. □

Lemma 69. *If* $G_1 \sqcap G_2 = G_3$*, then* $height(G_3) \leq max(height(G_1), height(G_2))$

Proof. By induction on $G_1 \sqcap G_2 = G_3$.

Case $(B \sqcap B = B)$. Trivial.

Case $(G \sqcap ? = ? \sqcap G = G)$. We know that $height(G) \geq 1$, and that $height(?) = 1$, therefore $max(height(G), height(?)) = height(G)$, and the result holds immediately.

Case $((G_{11} \rightarrow G_{12}) \sqcap (G_{21} \rightarrow G_{22}) = (G_{11} \sqcap G_{21}) \rightarrow (G_{12} \sqcap G_{22}))$. We know by induction hypothesis that $height(G_{11} \sqcap G_{21}) \leq max(height(G_{11}), height(G_{21}))$, and $height(G_{12} \sqcap G_{22}) \leq max(height(G_{12}), height(G_{22}))$. We also know that $height(G_{11} \rightarrow G_{12}) = 1 + max(height(G_{11}), height(G_{12}))$, and $height(G_{21} \rightarrow G_{22}) = 1 + max(height(G_{21}), height(G_{22}))$.

Then we have to prove that

$$height((G_{11} \sqcap G_{21}) \rightarrow (G_{12} \sqcap G_{22})) \leq max(height(G_{11} \rightarrow G_{12}), height(G_{21} \rightarrow G_{22}))$$

But we know that

$$\begin{aligned} & height((G_{11} \sqcap G_{21}) \rightarrow (G_{12} \sqcap G_{22})) \\ =& 1 + max(height(G_{11} \sqcap G_{21}), height(G_{12} \sqcap G_{22})) \\ \leq& 1 + max(max(height(G_{11}), height(G_{21})), max(height(G_{12}), height(G_{22}))) \end{aligned}$$

and that

$$\begin{aligned} & max(height(G_{11} \rightarrow G_{12}), height(G_{21} \rightarrow G_{22})) \\ =& max(1 + max(height(G_{11}), height(G_{12})), 1 + max(height(G_{21}), height(G_{22}))) \\ =& 1 + max(max(height(G_{11}), height(G_{12})), max(height(G_{21}), height(G_{22}))) \end{aligned}$$

therefore the result follows.

Case *(*$\mathsf{Ref}\ G_1 \sqcap \mathsf{Ref}\ G_2 = \mathsf{Ref}\ G_1 \sqcap G_2$*).* We know by induction hypothesis that $height(G_1 \sqcap G_2) \leq max(height(G_1), height(G_2))$. We also know that $height(\mathsf{Ref}\ G_1) = 1 + max(height(G_1), height(G_{12}))$. Then we have to prove that

$$height(\mathsf{Ref}\ G_1 \sqcap G_2) \leq max(height(\mathsf{Ref}\ G_1), height(\mathsf{Ref}\ G_2))$$

But we know that

$$\begin{aligned} height(\mathsf{Ref}\ G_1 \sqcap G_2) =& 1 + height(G_1 \sqcap G_2) \\ \leq& 1 + max(height(G_1), height(G_2)) \\ =& max(1 + height(G_1), 1 + height(G_2)) \\ =& max(height(\mathsf{Ref}\ G_1), height(\mathsf{Ref}\ G_2)) \end{aligned}$$

and the result holds. □

Lemma 70. *If* $\mathcal{I}_=(G_1, G_2) = \varepsilon$, *then* $height(\varepsilon) \leq max(height(G_1), height(G_2))$

Proof. Direct by 19 as $\langle G_1 \rangle \circ^= \langle G_2 \rangle = \langle G_1 \sqcap G_2 \rangle$ if defined. □

Lemma 71. *If* $\varepsilon_1 \circ^= \varepsilon_2 = \varepsilon_3$, *then* $height(\varepsilon_3) \leq max(height(\varepsilon_1), height(\varepsilon_2))$

Proof. Direct by 19 as $\langle G_1 \rangle \circ^= \langle G_2 \rangle = \langle G_1 \sqcap G_2 \rangle$ if defined. □

Lemma 72. *If* $\emptyset; \emptyset \vdash t \rightsquigarrow_n t : G$, *then if* ε *occurs in* t, *then* $\exists G'$ *in the derivation* $\emptyset; \emptyset \vdash t \rightsquigarrow_n t : G$, *such that* $height(\varepsilon) \leq height(G')$ *and* $size(\varepsilon) \leq 2^{height(G')} - 1$.

Proof. By induction on $\emptyset; \emptyset \vdash t \rightsquigarrow_n t : G$ using Lemmas 20 and 18. □

Lemma 73. *Let* $\varepsilon \vdash G_1 \sim G_2$, *then*

1. $height(idom(\varepsilon)) < height(\varepsilon)$ *if* $idom(\varepsilon)$ *is defined.*
2. $height(icod(\varepsilon)) < height(\varepsilon)$ *if* $icod(\varepsilon)$ *is defined.*
3. $height(iref(\varepsilon)) < height(\varepsilon)$ *if* $iref(\varepsilon)$ *is defined.*

Proposition 74. *If* $t \rightsquigarrow_n t : G$ *and* $t \mid \cdot \longmapsto^* t' \mid \mu'$ *such that* ε *occurs in* (t', μ'), *then there exists* G' *in the derivation of* $t \rightsquigarrow_n t : G$ *such that* $height(\varepsilon) \leq height(G')$ *and* $size(\varepsilon) \leq 2^{height(G')} - 1$.

Proof. By induction on the length of reduction $t \mid \cdot \longmapsto^* t' \mid \mu$.

Case *(*$t \mid \cdot \longmapsto^0 t \mid \cdot$*).* Direct by Lemma 72.

Case *(*$t \mid \cdot \longmapsto^k t'' \mid \mu''$, *and* $t'' \mid \mu'' \longmapsto t' \mid \mu',$*).* We only show representative cases. By induction hypothesis we know that, $\forall \varepsilon'$ such that ε' occurs in (t'', μ''), then $\exists G'$ in the derivation $\emptyset; \emptyset \vdash t \rightsquigarrow_n t : G$, such that $height(\varepsilon') \leq height(G')$ and $size(\varepsilon') \leq 2^{height(G')} - 1$ (1). One set of cases is when ε occurs in (t'', μ''), then the result follows immediately. Otherwise, the only cases that produces new evidences are the following:

- Case (RE) and (r2). Let $G_1 = G_{11} \rightarrow G_{12}$.

$$\varepsilon_1(\lambda x^{G_{11}}.t') \mathbin{@^{G_1 \rightarrow G}} \varepsilon_2 u \mid \mu'' \longmapsto icod(\varepsilon_1)([(\varepsilon_2 \circ^= idom(\varepsilon_1))u :: G_{11}/x^{G_{11}}]t') :: G_2 \mid \mu''$$

  By induction hypotheses, $\exists G'_1, G'_2, height(\varepsilon_1) \leq height(G'_1)$, and $height(\varepsilon_2) \leq height(G'_2)$. Let G' the tallest of types G'_1 and G'_2. Note that by Lemma 73, $height(idom(\varepsilon_1)) < height(\varepsilon_1)$, and $height(icod(\varepsilon_1)) < height(\varepsilon_1)$. Let $\varepsilon' = \varepsilon_2 \circ^= idom(\varepsilon_1)$. By Lemma 21, $height(\varepsilon') \leq \max(height(\varepsilon_2), height(idom(\varepsilon_1))) \leq \max(height(\varepsilon_2), height(\varepsilon_1)) \leq height(G')$. Then by Lemma 18, $size(\varepsilon') \leq 2^{height(\varepsilon')} - 1 \leq 2^{height(G')} - 1$. Also by Lemma 18, and as $height(icod(\varepsilon_1)) \leq height(\varepsilon_1) \leq height(G')$, then $size(icod(\varepsilon_1)) \leq 2^{height(icod(\varepsilon_1))} - 1 \leq 2^{height(G')} - 1$, and the result holds.
- Case (RE) and (r5). Analogously to (r2), noticing that by Lemma 73, $height(iref(\varepsilon)) \leq height(\varepsilon)$.
- Case (RE) and (r6).

$$\varepsilon_1 o^{G_1} :=^{G_3} \varepsilon_2 u \mid \mu'' \longmapsto \mathsf{unit} \mid \mu''[o^G \mapsto (\varepsilon_2 \circ^= iref(\varepsilon_1))u :: G_1]$$

By induction hypotheses, $\exists G_1', G_2', \mathit{height}(\varepsilon_1) \leq \mathit{height}(G_1')$, and $\mathit{height}(\varepsilon_2) \leq \mathit{height}(G_2')$. Let G' the tallest of types G_1' and G_2'. Note that by Lemma 73, $\mathit{height}(\mathit{iref}(\varepsilon_1)) < \mathit{height}(\varepsilon_1)$. Let $\varepsilon' = \varepsilon_2 \circ^{=} \mathit{iref}(\varepsilon_1)$. By Lemma 21, $\mathit{height}(\varepsilon') \leq \max(\mathit{height}(\varepsilon_2), \mathit{height}(\mathit{iref}(\varepsilon_1))) \leq \max(\mathit{height}(\varepsilon_2), \mathit{height}(\varepsilon_1)) \leq \mathit{height}(G')$. Then by Lemma 18, $\mathit{size}(\varepsilon') \leq 2^{\mathit{height}(\varepsilon')} - 1 \leq 2^{\mathit{height}(G')} - 1$, and the result holds.

- Case (RF) and (r7). Then $\varepsilon_1 t_1 \mid \mu'' \longmapsto \varepsilon_1' t_1'' \mid \mu''$. Suppose $\varepsilon_1 t_1 = \varepsilon_1(\varepsilon_1'' t_1'' :: G_1)t_1$. Then $\varepsilon_1' = \varepsilon_1 \circ^{=} \varepsilon_1''$. By Lemma 21, $\mathit{height}(\varepsilon_1') \leq \max(\mathit{height}(\varepsilon_1), \mathit{height}(\varepsilon_1''))$, but by (1) we know that $\exists G'$ in the elaboration, such that $\mathit{height}(\varepsilon_1) \leq \mathit{height}(G')$ and $\mathit{height}(\varepsilon_1'') \leq \mathit{height}(G')$. Therefore $\mathit{height}(\varepsilon_1') \leq \mathit{height}(G')$. By Lemma 18, $\mathit{size}(\varepsilon_1') \leq 2^{\mathit{height}(\varepsilon_1')} - 1$ but as $\mathit{height}(\varepsilon_1') \leq \mathit{height}(G')$, then $\mathit{size}(\varepsilon_1') \leq 2^{\mathit{height}(G')} - 1$ and the result follows. □

Proposition 75. *If $t \leadsto_n t : G$ and $t \mid \cdot \longmapsto^* t' \mid \mu'$, then there exists G' in the derivation of $t \leadsto_n t : G$ such that $\mathit{size}(\langle t', \mu' \rangle) \in O(2^{\mathit{height}(G')} \cdot \mathit{size}_{OR}(\langle t', \mu' \rangle))$.*

Proof. Rule (r7) prevents nesting of adjacent ascriptions in any term in the evaluation context, redex or store. Therefore the number of evidences of a program is proportional to the size of the program state (in the worst case, although assignments and applications introduce two evidences, each correspond to each of its subterms). Therefore by Proposition 22, the size of each evidence is in $O(2^{\mathit{height}(G')})$ for some G' in the elaboration of t. □

Appendix E. Relation to the coercion calculus

Lemma 76. $G_1 \sqcap G_2 = G \iff G_2 \sqcap G_1 = G$.

Proof. We prove both sides of the proposition by induction on the premise, i.e. by induction on $G_1 \sqcap G_2$ for the $\Rightarrow$ case, and induction on $G_2 \sqcap G_1$ for the $\Leftarrow$ (both cases as identical). □

Lemma 77. $\varepsilon_1 \circ^{=} \varepsilon_2 = \varepsilon \iff \varepsilon_2 \circ^{=} \varepsilon_1 = \varepsilon$

Proof. Direct by Proposition 76. □

Lemma 78. $\mathbf{c} = (\!|\varepsilon \vdash G_1 \sim G_2|\!)$*, then* nm $\mathbf{c}$

Proof. Straightforward induction on judgment $\varepsilon \vdash G_1 \sim G_2$. □

Proposition 79. *Let $\mathbf{c_1} = (\!|\varepsilon_1 \vdash G_1 \sim G_2|\!)$ and $\mathbf{c_2} = (\!|\varepsilon_2 \vdash G_2 \sim G_3|\!)$. Then*

1. $\mathbf{c_1}; \mathbf{c_2} \longmapsto^* \mathsf{Fail} \iff \varepsilon_1 \circ^{=} \varepsilon_2$ *is undefined*
2. $(\mathbf{c_1}; \mathbf{c_2} \longmapsto^* \mathbf{c} \wedge \text{nm}\ \mathbf{c}) \iff \varepsilon_1 \circ^{=} \varepsilon_2$ *is defined. Furthermore* $\mathbf{c} = (\!|(\varepsilon_1 \circ^{=} \varepsilon_2) \vdash G_1 \sim G_3|\!)$.

Proof. Direct by induction on types G_1, G_2 and G_3, inspection on the coercion reduction rules and transitivity of evidence. We only present interesting cases.

Case $(G_1 = R_1, G_2 = ?, G_3 = R_2)$. Then $\varepsilon_1 = \langle R_1 \rangle, \varepsilon_2 = \langle R_2 \rangle$, $\mathbf{c_1} = \mathbf{R_1}!$, and $\mathbf{c_2} = \mathbf{R_2}?$. If $R_1 \neq R_2$ then $R_1 \sqcap R_2$ is not defined and then $\varepsilon_1 \circ^{=} \varepsilon_2$ is not defined. Also $\mathbf{R_1}!; \mathbf{R_2}? \longmapsto \mathsf{Fail}$ and the result holds.
If $R_1 = R_2 = R$ then $R_1 \sqcap R_2 = R$ and then $\varepsilon_1 \circ^{=} \varepsilon_2 = \langle R \rangle$. Also $\mathbf{R}!; \mathbf{R}? \longmapsto \iota_{\mathbf{R}}$, but $\iota_{\mathbf{R}} = (\!|\langle R \rangle \vdash R \sim R|\!)$ and the result holds.

Case $(G_1 = ?, G_2 = R, G_3 = ?)$. Then $\varepsilon_1 = \langle R \rangle, \varepsilon_2 = \langle R \rangle$, $\mathbf{c_1} = \mathbf{R}?$, and $\mathbf{c_2} = \mathbf{R}!$. As $R \sqcap R = R$ then $\varepsilon_1 \circ^{=} \varepsilon_2 = \langle R \rangle$. Also $\mathbf{R}?; \mathbf{R}!$ is in normal form, but $\mathbf{R}?; \mathbf{R}! = (\!|\langle R \rangle \vdash ? \sim ?|\!)$ and the result holds.

Case $(G_1 = ?, G_2 = ?, G_3 = ?)$. Then we proceed on cases for ε_1 and ε_2.

1. $(\varepsilon_1 = \langle ? \rangle, \varepsilon_2 = \langle ? \rangle)$. Then $\mathbf{c_1} = \iota_?$ and $\mathbf{c_2} = \iota_?$. But $\iota_?; \iota_? \longrightarrow \iota_?$, nm $\iota_?$, $\varepsilon_1 \circ^{=} \varepsilon_2 = \langle ? \rangle$, and $\iota_? = (\!|\langle ? \rangle \vdash ? \sim ?|\!)$ and the result holds.
2. $(\varepsilon_1 = \langle R \rangle, \varepsilon_2 = \langle ? \rangle)$. Then $\mathbf{c_1} = \mathbf{R}?; \mathbf{R}!$ and $\mathbf{c_2} = \iota_?$. But $\mathbf{R}?; \mathbf{R}!; \iota_? \longrightarrow \mathbf{R}?; \mathbf{R}!$, nm $\mathbf{R}?; \mathbf{R}!$, $\varepsilon_1 \circ^{=} \varepsilon_2 = \langle R \rangle$, and $\mathbf{R}?; \mathbf{R}! = (\!|\langle R \rangle \vdash ? \sim ?|\!)$ and the result holds.
3. $(\varepsilon_1 = \langle ? \rangle, \varepsilon_2 = \langle R \rangle)$. Analogous to previous sub-case.
4. $(\varepsilon_1 = \langle R_1 \rangle, \varepsilon_2 = \langle R_2 \rangle)$. Then $\mathbf{c_1} = \mathbf{R_1}?; \mathbf{R_1}!$ and $\mathbf{c_2} = \mathbf{R_2}?; \mathbf{R_2}!$. If $R_1 \neq R_2$, then $\mathbf{R_1}?; \mathbf{R_1}!; \mathbf{R_2}?; \mathbf{R_2}! \longrightarrow^3 \mathsf{Fail}$, but also $\varepsilon_1 \circ^{=} \varepsilon_2 = \langle R_1 \rangle \circ^{=} \langle R_2 \rangle$ is undefined as $R_1 \sqcap R_2$ is not defined.

If $R_1 = R_2 = R$, then

$$\begin{aligned} &\mathbf{R_1}?; \mathbf{R_1}!; \mathbf{R_2}?; \mathbf{R_2}! \\ =&\mathbf{R}?; \mathbf{R}!; \mathbf{R}?; \mathbf{R}! \\ \longrightarrow&\mathbf{R}?; \iota_{\mathbf{R}}; \mathbf{R}! \\ \longrightarrow&\mathbf{R}?; \mathbf{R}! \\ \longrightarrow&\iota_{\mathbf{R}} \end{aligned}$$

where nm $\iota_{\mathbf{R}}$. But also $\varepsilon_1 \circ^{=} \varepsilon_2 = \langle R_1\rangle \circ^{=} \langle R_2\rangle = \langle R\rangle \circ^{=} \langle R\rangle = \langle R\rangle$, and $\iota_{\mathbf{R}} = (\!|\langle R\rangle \vdash ? \sim ?|\!)$ and the result holds.

Case *($G_1 = R_1, G_2 = ?, G_3 = ?$).* Then we proceed on cases for ε_2.

1. $(\varepsilon_1 = \langle R_1\rangle, \varepsilon_2 = \langle ?\rangle)$. Then $\mathbf{c_1} = \mathbf{R_1}?; \mathbf{R_1}!$ and $\mathbf{c_2} = \iota_?$. But $\mathbf{R_1}?; \mathbf{R_1}!; \iota_? \longrightarrow \mathbf{R_1}?; \mathbf{R_1}!$, nm $\mathbf{R_1}?; \mathbf{R_1}!$, $\varepsilon_1 \circ^{=} \varepsilon_2 = \langle R_1\rangle$, and $\mathbf{R_1}?; \mathbf{R_1}! = (\!|\langle R_1\rangle \vdash ? \sim ?|\!)$ and the result holds.
2. $(\varepsilon_1 = \langle R_1\rangle, \varepsilon_2 = \langle R_2\rangle)$. Then $\mathbf{c_1} = \mathbf{R_1}?; \mathbf{R_1}!$ and $\mathbf{c_2} = \mathbf{R_2}?; \mathbf{R_2}!$. If $R_1 \neq R_2$, then $\mathbf{R_1}?; \mathbf{R_1}!; \mathbf{R_2}?; \mathbf{R_2}! \longrightarrow^3$ Fail, but also $\varepsilon_1 \circ^{=} \varepsilon_2 = \langle R_1\rangle \circ^{=} \langle R_2\rangle$ is undefined as $R_1 \sqcap R_2$ is not defined.
   If $R_1 = R_2 = R$, then

$$\begin{aligned} &\mathbf{R_1}?; \mathbf{R_1}!; \mathbf{R_2}?; \mathbf{R_2}! \\ =\;&\mathbf{R}?; \mathbf{R}!; \mathbf{R}?; \mathbf{R}! \\ \longrightarrow&\mathbf{R}?; \iota_{\mathbf{R}}; \mathbf{R}! \\ \longrightarrow&\mathbf{R}?; \mathbf{R}! \\ \longrightarrow&\iota_{\mathbf{R}} \end{aligned}$$

   where nm $\iota_{\mathbf{R}}$. But also $\varepsilon_1 \circ^{=} \varepsilon_2 = \langle R_1\rangle \circ^{=} \langle R_2\rangle = \langle R\rangle \circ^{=} \langle R\rangle = \langle R\rangle$, and $\iota_{\mathbf{R}} = (\!|\langle R\rangle \vdash ? \sim ?|\!)$ and the result holds.

Case *($G_1 = ?, G_2 = ?, G_3 = R_2$).* Analogous to previous case.

Case *($G_1 = \text{Ref } G'_1 \neq R_1, G_2 = \text{Ref } G'_2 \neq R_2, G_3 = \text{Ref } G'_3 \neq R_3$).* Then $\varepsilon_1 = \langle \text{Ref } G'_{12}\rangle, \varepsilon_2 = \langle \text{Ref } G'_{23}\rangle$, $\mathbf{c_1} = \text{Ref } \mathbf{c_{21}}\ \mathbf{c_{12}}$, where $\mathbf{c_{21}} = (\!|\langle G'_{12}\rangle \vdash G'_2 \sim G'_1|\!)$ and $\mathbf{c_{12}} = (\!|\langle G'_{12}\rangle \vdash G'_1 \sim G'_2|\!)$, and $\mathbf{c_2} = \text{Ref } \mathbf{c_{32}}\ \mathbf{c_{23}}$, where $\mathbf{c_{32}} = (\!|\langle G'_{23}\rangle \vdash G'_3 \sim G'_2|\!)$ and $\mathbf{c_{23}} = (\!|\langle G'_{23}\rangle \vdash G'_2 \sim G'_3|\!)$.

But,

$$\begin{aligned} \mathbf{c_1}; \mathbf{c_2} = &\ (\text{Ref } \mathbf{c_{21}}\ \mathbf{c_{12}}); (\text{Ref } \mathbf{c_{32}}\ \mathbf{c_{23}}) \\ \longrightarrow&\text{Ref } (\mathbf{c_{32}}; \mathbf{c_{21}})\ (\mathbf{c_{12}}; \mathbf{c_{23}}) \end{aligned}$$

and $\varepsilon_1 \circ^{=} \varepsilon_2 = \langle \text{Ref } G'_{12}\rangle \circ^{=} \langle \text{Ref } G'_{23}\rangle = \langle \text{Ref } G'_{23}\rangle \circ^{=} \langle \text{Ref } G'_{12}\rangle$ (Proposition 77).

By induction hypothesis on $\mathbf{c_{32}} = (\!|\langle G'_{23}\rangle \vdash G'_3 \sim G'_2|\!)$ and $\mathbf{c_{21}} = (\!|\langle G'_{12}\rangle \vdash G'_2 \sim G'_1|\!)$, if $\mathbf{c_{32}}; \mathbf{c_{21}} \longrightarrow^* \text{Fail}$, and $G'_{23} \sqcap G'_{12}$ is not defined, therefore $\text{Ref } (\mathbf{c_{32}}; \mathbf{c_{21}})\ (\mathbf{c_{12}}; \mathbf{c_{23}}) \longrightarrow^* \text{Fail}$, and $\langle \text{Ref } G'_{12}\rangle \circ^{=} \langle \text{Ref } G'_{23}\rangle$ is not defined and the result holds. Similarly by induction hypothesis on $\mathbf{c_{12}} = (\!|\langle G'_{12}\rangle \vdash G'_1 \sim G'_2|\!)$ and $\mathbf{c_{23}} = (\!|\langle G'_{23}\rangle \vdash G'_2 \sim G'_3|\!)$, if $\mathbf{c_{32}}; \mathbf{c_{21}} \longrightarrow^* \text{Fail}$ and $G'_{12} \sqcap G'_{23}$ is not defined, therefore the result holds.

The only case left is that if we apply both induction hypotheses and we know that $\mathbf{c_{32}}; \mathbf{c_{21}} \longrightarrow^* \mathbf{c_{31}}$, nm $\mathbf{c_{31}}$, $\mathbf{c_{31}} = (\!|\langle G'_{23} \sqcap G'_{12}\rangle \vdash G'_3 \sim G'_1|\!) = (\!|\langle G'_{12} \sqcap G'_{23}\rangle \vdash G'_3 \sim G'_1|\!)$ (Proposition 77), $\mathbf{c_{12}}; \mathbf{c_{23}} \longrightarrow^* \mathbf{c_{13}}$, nm $\mathbf{c_{13}}$, and $\mathbf{c_{13}} = (\!|\langle G'_{12} \sqcap G'_{23}\rangle \vdash G'_1 \sim G'_3|\!)$. Then $\text{Ref } (\mathbf{c_{32}}; \mathbf{c_{21}})\ (\mathbf{c_{12}}; \mathbf{c_{23}}) \longrightarrow^* \text{Ref } \mathbf{c_{31}}\ \mathbf{c_{13}}$, nm $\text{Ref } \mathbf{c_{31}}\ \mathbf{c_{13}}$, and $\varepsilon_1 \circ^{=} \varepsilon_2 = \langle \text{Ref } G'_{12}\rangle \circ^{=} \langle \text{Ref } G'_{23}\rangle = \langle \text{Ref } G'_{12} \sqcap G'_{23}\rangle$.

But $(\!|\langle \text{Ref } G'_{12} \sqcap G'_{23}\rangle \vdash \text{Ref } G'_1 \sim \text{Ref } G'_2|\!) = \text{Ref } (\!|\langle \mathbf{G'_{12}} \sqcap \mathbf{G'_{23}}\rangle \vdash \mathbf{G'_3} \sim \mathbf{G'_1}|\!)\ (\!|\langle \mathbf{G'_{12}} \sqcap \mathbf{G'_{23}}\rangle \vdash \mathbf{G'_1} \sim \mathbf{G'_3}|\!) = \text{Ref } \mathbf{c_{31}}\ \mathbf{c_{13}}$ and the result holds.

Case *($G_1 = \text{Ref } G'_1 \neq R_1, G_2 = \text{Ref } G'_2 \neq R_2, G_3 = ?$).* Then $\varepsilon_1 = \langle \text{Ref } G'_{12}\rangle, \varepsilon_2 = \langle \text{Ref } G'_{23}\rangle$, $\mathbf{c_1} = \text{Ref } \mathbf{c_{21}}\ \mathbf{c_{12}}$, where $\mathbf{c_{21}} = (\!|\langle G'_{12}\rangle \vdash G'_2 \sim G'_1|\!)$ and $\mathbf{c_{12}} = (\!|\langle G'_{12}\rangle \vdash G'_1 \sim G'_2|\!)$, and $\mathbf{c_2} = \text{Ref } \mathbf{c_{32}}\ \mathbf{c_{23}}$, where $\mathbf{c_{32}} = (\!|\langle G'_{23}\rangle \vdash G'_3 \sim G'_2|\!)$ and $\mathbf{c_{23}} = (\!|\langle G'_{23}\rangle \vdash G'_2 \sim G'_3|\!)$, and we proceed analogous to previous case.

Case *($G_1 = \text{Ref } G'_1 \neq R_1, G_2 = ?, G_3 = ?$).* Then $\varepsilon_1 = \langle \text{Ref } G'_{12}\rangle$, $\mathbf{c_1} = (\!|\langle \text{Ref } G'_{12}\rangle \vdash \text{Ref } G'_1 \sim \text{Ref } ?|\!); (\text{Ref } ?)!$, and we proceed by cases for ε_2.

1. $(\varepsilon_2 = \langle ?\rangle)$. Then $\mathbf{c_2} = \iota_?$, and $\mathbf{c_1}; \mathbf{c_2} \longrightarrow \mathbf{c_1}$. We also know nm $\mathbf{c_1}$ (by Lemma 78), and $\varepsilon_1 \circ^{=} \varepsilon_2 = \langle \text{Ref } G'_{12} \sqcap ?\rangle = \langle \text{Ref } G'_{12}\rangle$, but $\mathbf{c_1} = (\!|\langle \text{Ref } G'_{12}\rangle \vdash \text{Ref } G'_1 \sim ?|\!)$, and the result holds.

2. $(\varepsilon_2 = \langle \text{Ref } ? \rangle)$. Then $\mathbf{c_2} = (\text{Ref } ?)?; (\text{Ref } ?)!$, and

$$\begin{aligned}\mathbf{c_1}; \mathbf{c_2} &= (\!|\langle \text{Ref } G'_{12}\rangle \vdash \text{Ref } G'_1 \sim \text{Ref } ?|\!); (\text{Ref } ?)!; (\text{Ref } ?)?; (\text{Ref } ?)! \\ &\longrightarrow (\!|\langle \text{Ref } G'_{12}\rangle \vdash \text{Ref } G'_1 \sim \text{Ref } ?|\!); (\text{Ref } ?)! \\ &\longrightarrow (\!|\langle \text{Ref } G'_{12}\rangle \vdash \text{Ref } G'_1 \sim \text{Ref } ?|\!); \iota_{\text{Ref } ?}; (\text{Ref } ?)! \\ &= \mathbf{c_1}\end{aligned}$$

i.e. $\mathbf{c_1}; \mathbf{c_2} \longrightarrow^* \mathbf{c_1}$. We also know nm $\mathbf{c_1}$ (by Lemma 78), and $\varepsilon_1 \circ^= \varepsilon_2 = \langle (\text{Ref } G'_{12}) \sqcap (\text{Ref } ?)\rangle = \langle \text{Ref } (G'_{12} \sqcap ?)\rangle = \langle \text{Ref } G'_{12}\rangle$, but $\mathbf{c_1} = (\!|\langle \text{Ref } G'_{12}\rangle \vdash \text{Ref } G'_1 \sim ?|\!)$, and the result holds.

3. $(\varepsilon_2 = \langle R \rangle, R \neq \text{Ref } ?)$. Then $\mathbf{c_2} = \mathbf{R}?; \mathbf{R}!$, and

$$\begin{aligned}\mathbf{c_1}; \mathbf{c_2} &= (\!|\langle \text{Ref } G'_{12}\rangle \vdash \text{Ref } G'_1 \sim \text{Ref } ?|\!); (\text{Ref } ?)!; \mathbf{R}?; \mathbf{R}! \\ &\longrightarrow (\!|\langle \text{Ref } G'_{12}\rangle \vdash \text{Ref } G'_1 \sim \text{Ref } ?|\!); \text{Fail} \\ &\longrightarrow^* \text{Fail}\end{aligned}$$

We also know that $\varepsilon_1 \circ^= \varepsilon_2 =$ is not defined as $\text{Ref } G'_{12}) \sqcap (R)$ is undefined, and the result holds immediately. □

Lemma 80. *If $\varepsilon_1 \circ^= \varepsilon_2$ is not defined then $\forall \varepsilon'_2 \sqsubseteq \varepsilon_2$, $\varepsilon_1 \circ^= \varepsilon'_2$ is not defined.*

Proof. By induction on ε_1 and ε_2 subject to consistent transitivity being not defined. □

Lemma 81. *If $\varepsilon_1 \circ^= \varepsilon_2$ is not defined then $\forall \varepsilon'_1 \sqsubseteq \varepsilon_1$, $\varepsilon'_1 \circ^= \varepsilon_2$ is not defined.*

Proof. Direct by Lemma 80 and Lemma 77. □

Proposition 82. *If $G = G_1 \sqcap G_2$ is defined, then $G \sqsubseteq G_1$ and $G \sqsubseteq G_2$.*

Proof. Straightforward induction on $G_1 \sqcap G_2$. □

Proposition 83. *If $\langle \text{Ref } G\rangle \vdash \text{Ref } G_1 \sim \text{Ref } G_2$ then $G \vdash G_1 \sim G_2$.*

Proof. Straightforward induction on $\langle \text{Ref } G\rangle \vdash \text{Ref } G_1 \sim \text{Ref } G_2$. □

Proposition 84. *If $\langle G_1 \rightarrow G_2\rangle \vdash G_{11} \rightarrow G_{12} \sim G_{21} \rightarrow G_{22}$ then $G_1 \vdash G_{21} \sim G_{11}$.*

Proof. Straightforward induction on $\langle G_1 \rightarrow G_2\rangle \vdash G_{11} \rightarrow G_{12} \sim G_{21} \rightarrow G_{22}$. □

Proposition 85. *If $\langle G_1 \rightarrow G_2\rangle \vdash G_{11} \rightarrow G_{12} \sim G_{21} \rightarrow G_{22}$ then $G_2 \vdash G_{12} \sim G_{22}$.*

Proof. Straightforward induction on $\langle \text{Ref } G\rangle \vdash \text{Ref } G_1 \sim \text{Ref } G_2$. □

Proposition 86 *(Optimality). If $\varepsilon = \varepsilon_1 \circ^= \varepsilon_2$ is defined, then $\pi_1(\varepsilon) \sqsubseteq \pi_1(\varepsilon_1)$ and $\pi_2(\varepsilon) \sqsubseteq \pi_2(\varepsilon_2)$.*

Proof. Direct by Lemma 82, as evidences can be represented as singletons. □

Lemma 87. $(\mathbf{c_1}; \mathbf{c_2} \longmapsto^* \mathbf{c} \wedge \mathbf{c}; \mathbf{c_3} \longmapsto^* \mathbf{c}') \iff (\mathbf{c_1}; \mathbf{c_2}); \mathbf{c_3} \longmapsto^* \mathbf{c}'$

Proof. Straightforward induction on $(\mathbf{c_1}; \mathbf{c_2})$ and then induction on $\mathbf{c_3}$. □

Lemma 88. $(\mathbf{c_2}; \mathbf{c_3} \longmapsto^* \mathbf{c} \wedge \mathbf{c_1}; \mathbf{c} \longmapsto^* \mathbf{c}') \iff (\mathbf{c_1}; \mathbf{c_2}); \mathbf{c_3} \longmapsto^* \mathbf{c}'$

Proof. For proving $(\Rightarrow)$, we use straightforward induction on $(\mathbf{c_2}; \mathbf{c_3})$ and then induction on $\mathbf{c_1}$. For proving the other direction $(\Leftarrow)$, we use induction on $(\mathbf{c_1}; \mathbf{c_2}); \mathbf{c_3}$. □

Proposition 89 *(Associativity). Let $\varepsilon_1 \vdash G_1 \sim G_2$, $\varepsilon_2 \vdash G_2 \sim G_3$ and $\varepsilon_3 \vdash G_3 \sim G_4$. Then $(\varepsilon_1 \circ^= \varepsilon_2) \circ^= \varepsilon_3 = \varepsilon_1 \circ^= (\varepsilon_2 \circ^= \varepsilon_3)$ or both are undefined.*

Proof. By straightforward induction on evidences $\varepsilon_1, \varepsilon_2$, and ε_3, noticing that if $\varepsilon_1 = \langle G_{12}\rangle, \varepsilon_2 = \langle G_{23}\rangle, \varepsilon_3 = \langle G_{34}\rangle$, then it is equivalent to prove that $(G_{12} \sqcap G_{23}) \sqcap G_{34} = G_{12} \sqcap (G_{23} \sqcap G_{34})$ or both are undefined. We only present interesting cases.

Case *($\varepsilon_1 = \langle G\rangle, \varepsilon_1 = \langle G\rangle, \varepsilon_1 = \langle G\rangle$).* Then the result is trivial as $\langle G\rangle \circ^= \langle G\rangle = \langle G\rangle$.

Case *($\varepsilon_1 = \langle G_1\rangle, \varepsilon_1 = \langle ?\rangle, \varepsilon_1 = \langle G_2\rangle$).* As $(\varepsilon_1 \circ^= \varepsilon_2) \circ^= \varepsilon_3 = (\langle G_1\rangle \circ^= \langle ?\rangle) \circ^= \langle G_2\rangle = \langle G_1\rangle \circ^= \langle G_2\rangle$, and $\varepsilon_1 \circ^= (\varepsilon_2 \circ^= \varepsilon_3) = \langle G_1\rangle \circ^= (\langle ?\rangle \circ^= \langle G_2\rangle) = \langle G_1\rangle \circ^= \langle G_2\rangle$, the result holds immediately.

Case *($\varepsilon_1 = \langle G\rangle, \varepsilon_1 = \langle ?\rangle, \varepsilon_1 = \langle ?\rangle$).* As $(\varepsilon_1 \circ^= \varepsilon_2) \circ^= \varepsilon_3 = (\langle G\rangle \circ^= \langle ?\rangle) \circ^= \langle ?\rangle = \langle G\rangle \circ^= \langle G\rangle = \langle G\rangle$, and $\varepsilon_1 \circ^= (\varepsilon_2 \circ^= \varepsilon_3) = \langle G\rangle \circ^= (\langle ?\rangle \circ^= \langle ?\rangle) = \langle G\rangle \circ^= \langle ?\rangle = \langle B\rangle$, the result holds immediately.

Case *($\varepsilon_1 = \langle ?\rangle, \varepsilon_1 = \langle ?\rangle, \varepsilon_1 = \langle ?\rangle$).* As $(\varepsilon_1 \circ^= \varepsilon_2) \circ^= \varepsilon_3 = (\langle ?\rangle \circ^= \langle ?\rangle) \circ^= \langle ?\rangle = \langle ?\rangle \circ^= \langle ?\rangle = \langle ?\rangle$, and $\varepsilon_1 \circ^= (\varepsilon_2 \circ^= \varepsilon_3) = \langle ?\rangle \circ^= (\langle ?\rangle \circ^= \langle ?\rangle) = \langle ?\rangle \circ^= \langle ?\rangle = \langle ?\rangle$, the result holds immediately.

Case *($\varepsilon_1 = \langle G_1\rangle, \varepsilon_1 = \langle G_2\rangle, \varepsilon_1 = \langle ?\rangle$).* As $(\varepsilon_1 \circ^= \varepsilon_2) \circ^= \varepsilon_3 = (\langle G_1\rangle \circ^= \langle G_2\rangle) \circ^= \langle ?\rangle = \langle G_1 \sqcap G_2\rangle \circ^= \langle ?\rangle = \langle G_1 \sqcap G_2\rangle$ if $G_1 \sqcap G_2$ is defined, and $\varepsilon_1 \circ^= (\varepsilon_2 \circ^= \varepsilon_3) = \langle G_1\rangle \circ^= (\langle G_2\rangle \circ^= \langle ?\rangle) = \langle G_1\rangle \circ^= \langle G_2\rangle = \langle G_1 \sqcap G_2\rangle$ if $G_1 \sqcap G_2$ is defined, the result holds immediately.

Case *($\varepsilon_1 = \langle ?\rangle, \varepsilon_1 = \langle G\rangle, \varepsilon_1 = \langle ?\rangle$).* As $(\varepsilon_1 \circ^= \varepsilon_2) \circ^= \varepsilon_3 = (\langle ?\rangle \circ^= \langle G\rangle) \circ^= \langle ?\rangle = \langle G\rangle \circ^= \langle ?\rangle = \langle G\rangle$, and $\varepsilon_1 \circ^= (\varepsilon_2 \circ^= \varepsilon_3) = \langle ?\rangle \circ^= (\langle G\rangle \circ^= \langle ?\rangle) = \langle ?\rangle \circ^= \langle G\rangle = \langle G\rangle$, the result holds immediately.

Case *($\varepsilon_1 = \langle G_1\rangle, \varepsilon_1 = \langle G_2\rangle, \varepsilon_1 = \langle G_3\rangle$, $G_1 \sqcap G_2$ not defined).* We know that $(\varepsilon_1 \circ^= \varepsilon_2) \circ^= \varepsilon_3 = (\langle G_1\rangle \circ^= \langle G_2\rangle) \circ^= \langle G_3\rangle$ is not defined, and that $\varepsilon_1 \circ^= (\varepsilon_2 \circ^= \varepsilon_3) = \langle G_1\rangle \circ^= (\langle G_2\rangle \circ^= \langle G_3\rangle) = \langle G_1\rangle \circ^= \langle G_2 \sqcap G_3\rangle$ (if $G_2 \sqcap G_3$ is not defined the result holds immediately). By Proposition 86, $\langle G_2 \sqcap G_3\rangle \sqsubseteq \langle G_2\rangle$. Then by Proposition 80 $\langle G_1\rangle \circ^= \langle G_2 \sqcap G_3\rangle$ is not defined, and the result holds.

Case *($\varepsilon_1 = \langle G_1\rangle, \varepsilon_1 = \langle G_2\rangle, \varepsilon_1 = \langle G_3\rangle$, $G_2 \sqcap G_3$ not defined).* We know that $\varepsilon_1 \circ^= (\varepsilon_2 \circ^= \varepsilon_3) = \langle G_1\rangle \circ^= (\langle G_2\rangle \circ^= \langle G_3\rangle)$ is not defined, and that $(\varepsilon_1 \circ^= \varepsilon_2) \circ^= \varepsilon_3 = (\langle G_1\rangle \circ^= \langle G_2\rangle) \circ^= \langle G_3\rangle = \langle G_1 \sqcap G_2\rangle \circ^= \langle G_3\rangle$ (if $G_1 \sqcap G_2$ is not defined the result holds immediately). By Proposition 86, $\langle G_1 \sqcap G_2\rangle \sqsubseteq \langle G_2\rangle$. Then by Proposition 81 $\langle G_1 \sqcap G_2\rangle \circ^= \langle G_3\rangle$ is not defined, and the result holds.

Case *($\varepsilon_1 = \langle G_1\rangle, \varepsilon_1 = \langle G_2\rangle, \varepsilon_1 = \langle G_3\rangle$, $G_1 \sqcap G_3$ not defined).* We know that $(\varepsilon_1 \circ^= \varepsilon_2) \circ^= \varepsilon_3 = (\langle G_1 \sqcap G_2\rangle) \circ^= \langle G_3\rangle$, and that $\varepsilon_1 \circ^= (\varepsilon_2 \circ^= \varepsilon_3) = \langle G_1\rangle \circ^= (\langle G_2 \sqcap G_3\rangle)$ (if $G_1 \sqcap G_2$ or $G_2 \sqcap G_3$ are not defined then the result holds immediately by next argument).

Then by Proposition 86, $\langle G_1 \sqcap G_2\rangle \sqsubseteq \langle G_1\rangle$ and $\langle G_2 \sqcap G_3\rangle \sqsubseteq \langle G_3\rangle$, therefore by Proposition 81 $\langle G_1 \sqcap G_2\rangle \circ^= \langle G_3\rangle$ is not defined, and by Proposition 80, $\langle G_1\rangle \circ^= \langle G_2 \sqcap G_3\rangle$ is not defined, and the result holds as both combinations of evidence fail, regardless if $G_1 \sqcap G_2$ or $G_2 \sqcap G_3$ are defined or not.

Case *($\varepsilon_1 = \langle \mathsf{Ref}\ G_1'\rangle, \varepsilon_1 = \langle \mathsf{Ref}\ G_2'\rangle, \varepsilon_1 = \langle \mathsf{Ref}\ G_3'\rangle$).* Notice that $(\langle \mathsf{Ref}\ G_1'\rangle \circ^= \langle \mathsf{Ref}\ G_2'\rangle) \circ^= \langle \mathsf{Ref}\ G_3'\rangle = \langle \mathsf{Ref}\ G\rangle$ is defined if and only if $\langle G\rangle = (\langle G_1'\rangle \circ^= \langle G_2'\rangle) \circ^= \langle G_3'\rangle$ is defined. Similarly $\langle \mathsf{Ref}\ G_1'\rangle \circ^= (\langle \mathsf{Ref}\ G_2'\rangle \circ^= \langle \mathsf{Ref}\ G_3'\rangle) = \langle \mathsf{Ref}\ G'\rangle$ is defined if and only if $\langle G'\rangle = \langle G_1'\rangle \circ^= (\langle G_2'\rangle \circ^= \langle G_3'\rangle)$ is defined. Also $\langle \mathsf{Ref}\ G_1'\rangle \vdash \mathsf{Ref}\ G_1'' \sim \mathsf{Ref}\ G_2''$, $\langle \mathsf{Ref}\ G_2'\rangle \vdash \mathsf{Ref}\ G_2'' \sim \mathsf{Ref}\ G_3''$, and $\langle \mathsf{Ref}\ G_3'\rangle \vdash \mathsf{Ref}\ G_3'' \sim \mathsf{Ref}\ G_4''$, then by inversion lemma (Lemma 83), $\langle G_1'\rangle \vdash G_1'' \sim G_2''$, $\langle G_2'\rangle \vdash G_2'' \sim G_3''$, and $\langle G_3'\rangle \vdash G_3'' \sim G_4''$

Then the result holds immediately as by induction hypothesis $(\langle G_1'\rangle \circ^= \langle G_2'\rangle) \circ^= \langle G_3'\rangle = \langle G_1'\rangle \circ^= (\langle G_2'\rangle \circ^= \langle G_3'\rangle)$ or both are undefined.

Case *($\varepsilon_1 = \langle G_{11}' \to G_{12}'\rangle, \varepsilon_1 = \langle G_{21}' \to G_{22}'\rangle, \varepsilon_1 = \langle G_{31}' \to G_{32}'\rangle$).* Analogous to the previous case but using inversion Lemmas 85 and 84. □

Proposition 90. *Let $\mathbf{c_1} \vdash G_1 \Rightarrow G_2$, $\mathbf{c_2} \vdash G_2 \Rightarrow G_3$ and $\mathbf{c_3} \vdash G_3 \Rightarrow G_4$. Then either*

- $(\mathbf{c_1}; \mathbf{c_2}); \mathbf{c_3} \longmapsto^* \mathbf{c}$ *and* $\mathbf{c_1}; (\mathbf{c_2}; \mathbf{c_3}) \longmapsto^* \mathbf{c}$, *or*
- $(\mathbf{c_1}; \mathbf{c_2}); \mathbf{c_3} \longmapsto^* \mathsf{Fail}$ *and* $\mathbf{c_1}; (\mathbf{c_2}; \mathbf{c_3}) \longmapsto^* \mathsf{Fail}$

Proof. By induction on $\mathbf{c_1}, \mathbf{c_2}$, and $\mathbf{c_3}$. Alternatively, by Proposition 79 and Proposition 89, noticing that $\mathbf{c_i} = (\!|\varepsilon_i \vdash G_i \sim G_{i+1}|\!)$ for some ε_i. □

Proposition 91 *(Confluence). Let $\mathbf{c_1} \vdash G_1 \Rightarrow G_2$, $\mathsf{nm}\ \mathbf{c_1}$, $\mathbf{c_2} \vdash G_2 \Rightarrow G_3$, and $\mathsf{nm}\ \mathbf{c_2}$. If $\mathbf{c_1}; \mathbf{c_2} \longmapsto^* \mathbf{c_1'}$ and $\mathsf{nm}\ \mathbf{c_1'}$, and $\mathbf{c_1}; \mathbf{c_2} \longmapsto^* \mathbf{c_2'}$ and $\mathsf{nm}\ \mathbf{c_1'}$, then $\mathbf{c_1'} = \mathbf{c_2'}$.*

Proof. By a lengthy induction on $\mathbf{c_1} \vdash G_1 \Rightarrow G_2 \wedge \mathsf{nm}\ \mathbf{c_1}$ and then induction on $\mathbf{c_2} \vdash G_2 \Rightarrow G_3 \wedge \mathsf{nm}\ \mathbf{c_2}$. We show a representative case:

Case $(\mathbf{c_1} = (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; (? \to ?)!)$. where $\mathsf{nm}\ \mathbf{c_{1i}}$, and $G_2 = ?$. We now proceed by induction on $\mathbf{c_2} \vdash ? \Rightarrow G_3 \wedge \mathsf{nm}\ \mathbf{c_2}$:

- (Case $\mathbf{c_2} = \mathsf{Fail}$). Then

$$\begin{aligned} (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; (? \to ?)!; \mathsf{Fail} &\longmapsto (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; \mathsf{Fail} && (\text{as } \mathsf{nm}\ (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}) \\ &\longmapsto (? \to ?)?; \mathsf{Fail} && (\text{as } \mathsf{nm}\ (? \to ?)?) \\ &\longmapsto \mathsf{Fail} \end{aligned}$$

  and as $\mathsf{nm}\ \mathsf{Fail}$, the result holds.
- (Case $\mathbf{c_2} = ? \to ??$). Then

$$\begin{aligned} (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; (? \to ?)!; ? \to ?? &\longmapsto (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; \iota_{? \to ?} && (\text{as } \mathsf{nm}\ (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}) \\ &\longmapsto (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}} && (\text{as } \mathsf{nm}\ (? \to ?)?) \end{aligned}$$

  and as $\mathsf{nm}\ (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}$, the result holds.
- (Case $\mathbf{c_2} = \mathsf{Ref}\ ??$). Then

$$\begin{aligned} (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; (? \to ?)!; \mathsf{Ref}\ ?? &\longmapsto (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; \mathsf{Fail} && (\text{as } \mathsf{nm}\ (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}) \\ &\longmapsto (? \to ?)?; \mathsf{Fail} && (\text{as } \mathsf{nm}\ (? \to ?)?) \\ &\longmapsto \mathsf{Fail} \end{aligned}$$

  and as $\mathsf{nm}\ \mathsf{Fail}$, the result holds.
- (Case $\mathbf{c_2} = \mathbf{B}?$). Similar to $\mathbf{c_2} = \mathsf{Ref}\ ??$ case.
- (Case $\mathbf{c_2} = (? \to ?)?; \mathbf{c_{21}} \to \mathbf{c_{22}}; (? \to ?)!$). Then

$$\begin{aligned} &(? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; (? \to ?)!; (? \to ?)?; \mathbf{c_{21}} \to \mathbf{c_{22}}; (? \to ?)! \\ \longmapsto &(? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; \iota_{? \to ?}; \mathbf{c_{21}} \to \mathbf{c_{22}}; (? \to ?)! \end{aligned}$$

  as $\mathsf{nm}\ (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}$ and $\mathsf{nm}\ \mathbf{c_{21}} \to \mathbf{c_{22}}; (? \to ?)!$. Then

$$\begin{aligned} &(? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; \iota_{? \to ?}; \mathbf{c_{21}} \to \mathbf{c_{22}}; (? \to ?)! \\ \longmapsto &(? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; \mathbf{c_{21}} \to \mathbf{c_{22}}; (? \to ?)! \end{aligned}$$

  Notice that we get to the same result either if we reduce the sub coercion $\mathbf{c_{11}} \to \mathbf{c_{12}}; \iota_{? \to ?}$, or $\iota_{? \to ?}; \mathbf{c_{21}} \to \mathbf{c_{22}}$. Then,

$$\begin{aligned} &(? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; \mathbf{c_{21}} \to \mathbf{c_{22}}; (? \to ?)! \\ \longmapsto &(? \to ?)?; (\mathbf{c_{21}}; \mathbf{c_{11}}) \to (\mathbf{c_{12}};; \mathbf{c_{22}}); (? \to ?)! \end{aligned}$$

  as $\mathsf{nm}\ ? \to ??$ and $\mathsf{nm}\ ? \to ?!$. Now we apply induction hypotheses:
  (1) if $\mathbf{c_{21}}; \mathbf{c_{11}} \longmapsto^* \mathbf{c'_{11}}$ and $\mathbf{c_{21}}; \mathbf{c_{11}} \longmapsto^* \mathbf{c'_{12}}$, $\mathsf{nm}\ \mathbf{c'_{11}}$, and $\mathsf{nm}\ \mathbf{c'_{12}}$, then $\mathbf{c'_{11}} = \mathbf{c'_{12}}$.
  (2) if $\mathbf{c_{12}}; \mathbf{c_{22}} \longmapsto^* \mathbf{c'_{21}}$ and $\mathbf{c_{12}}; \mathbf{c_{22}} \longmapsto^* \mathbf{c'_{22}}$, $\mathsf{nm}\ \mathbf{c'_{21}}$, and $\mathsf{nm}\ \mathbf{c'_{22}}$, then $\mathbf{c'_{21}} = \mathbf{c'_{22}}$.
  Then

$$\begin{aligned} &(? \to ?)?; (\mathbf{c_{21}}; \mathbf{c_{11}}) \to (\mathbf{c_{12}};; \mathbf{c_{22}}); (? \to ?)! \\ \longmapsto^* &(? \to ?)?; \mathbf{c'_{11}} \to \mathbf{c'_{12}}; (? \to ?)! \end{aligned}$$

  But as $\mathsf{nm}\ \mathbf{c'_{11}}$ and $\mathsf{nm}\ \mathbf{c'_{12}}$, then $\mathsf{nm}\ (? \to ?)?; \mathbf{c'_{11}} \to \mathbf{c'_{12}}; (? \to ?)!$ and the result follows.
- (Case $\mathbf{c_2} = (? \to ?)?; \mathbf{c_{21}} \to \mathbf{c_{22}}$). Then

$$\begin{aligned} &(? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; (? \to ?)!; (? \to ?)?; \mathbf{c_{21}} \to \mathbf{c_{22}} \\ \longmapsto &(? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; \iota_{? \to ?}; \mathbf{c_{21}} \to \mathbf{c_{22}} \end{aligned}$$

  as $\mathsf{nm}\ (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}$ and $\mathsf{nm}\ \mathbf{c_{21}} \to \mathbf{c_{22}}$. Then

$$(? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; \iota_{? \to ?}; \mathbf{c_{21}} \to \mathbf{c_{22}}$$
$$\longmapsto (? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; \mathbf{c_{21}} \to \mathbf{c_{22}}$$

Notice that we get to the same result either if we reduce the sub coercion $\mathbf{c_{11}} \to \mathbf{c_{12}}; \iota_{? \to ?}$, or $\iota_{? \to ?}; \mathbf{c_{21}} \to \mathbf{c_{22}}$. Then,

$$(? \to ?)?; \mathbf{c_{11}} \to \mathbf{c_{12}}; \mathbf{c_{21}} \to \mathbf{c_{22}}$$
$$\longmapsto (? \to ?)?; (\mathbf{c_{21}}; \mathbf{c_{11}}) \to (\mathbf{c_{12}}; ; \mathbf{c_{22}})$$

as nm $? \to ??$. Now we apply induction hypotheses:
(1) if $\mathbf{c_{21}}; \mathbf{c_{11}} \longmapsto^* \mathbf{c'_{11}}$ and $\mathbf{c_{21}}; \mathbf{c_{11}} \longmapsto^* \mathbf{c'_{12}}$, nm $\mathbf{c'_{11}}$, and nm $\mathbf{c'_{12}}$, then $\mathbf{c'_{11}} = \mathbf{c'_{12}}$.
(2) if $\mathbf{c_{12}}; \mathbf{c_{22}} \longmapsto^* \mathbf{c'_{21}}$ and $\mathbf{c_{12}}; \mathbf{c_{22}} \longmapsto^* \mathbf{c'_{22}}$, nm $\mathbf{c'_{21}}$, and nm $\mathbf{c'_{22}}$, then $\mathbf{c'_{21}} = \mathbf{c'_{22}}$.
Then

$$(? \to ?)?; (\mathbf{c_{21}}; \mathbf{c_{11}}) \to (\mathbf{c_{12}}; ; \mathbf{c_{22}})$$
$$\longmapsto^* (? \to ?)?; \mathbf{c'_{11}} \to \mathbf{c'_{12}}$$

But as nm $\mathbf{c'_{11}}$ and nm $\mathbf{c'_{12}}$, then nm $(? \to ?)?; \mathbf{c'_{11}} \to \mathbf{c'_{12}}$ and the result follows.

- (Case $\mathbf{c_2} = \text{Ref } ??; \text{Ref } \mathbf{c_{21}}\ \mathbf{c_{22}}; \text{Ref } ?!$). Analogous to the function case.
- (Case $\mathbf{c_2} = \text{Ref } ??; \text{Ref } \mathbf{c_{21}}\ \mathbf{c_{22}}$). Analogous to the function case. □

Lemma 92. *$\varepsilon_1 t_1 \ @^{G_1 \to G_2}\ \varepsilon_2 t_2 \mid \mu \longmapsto \varepsilon'_1 t'_1 \ @^{G_1 \to G_2}\ \varepsilon_2 t_2 \mid \mu'$ if and only if $\varepsilon_1 t_1 :: G_1 \to G_2 \mid \mu \longmapsto \varepsilon'_1 t'_1 :: G_1 \to G_2 \mid \mu'$.*

Proof. We start by proving $\Rightarrow$ by case analysis on t_1 (the other direction is analogous).

- If $t_1 = \varepsilon_3 t_3 :: G_3$ where $\varepsilon_1(\varepsilon_3 t_3 :: G_3) \ @^{G_1 \to G_2}\ \varepsilon_2 t_2 \mid \mu \longmapsto \varepsilon_1(\varepsilon'_3 t'_3 :: G_3) \ @^{G_1 \to G_2}\ \varepsilon_2 t_2 \mid \mu'$, then it is easy to see that $\varepsilon_1(\varepsilon_3 t_3 :: G_3) :: G_1 \to G_2 \mid \mu \longmapsto \varepsilon_1(\varepsilon'_3 t'_3 :: G_3) :: G_1 \to G_2 \mid \mu'$ and the result holds.
- If $t_1 = \varepsilon_3 u :: G_3$ where $\varepsilon_1(\varepsilon_3 u :: G_3) \ @^{G_1 \to G_2}\ \varepsilon_2 t_2 \mid \mu \longmapsto \varepsilon'_1 u \ @^{G_1 \to G_2}\ \varepsilon_2 t_2 \mid \mu$ and $\varepsilon'_1 = \varepsilon_3 \circ \varepsilon_1$, then also $\varepsilon_1(\varepsilon_3 u :: G_3) :: G_1 \to G_2 \mid \mu \longmapsto \varepsilon'_1 u :: G_1 \to G_2 \mid \mu$ and the result holds. □

Lemma 93. *$\varepsilon_1 t_1 \ @^{G_1 \to G_2}\ \varepsilon_2 t_2 \mid \mu \longmapsto \varepsilon_1 t_1 \ @^{G_1 \to G_2}\ \varepsilon'_2 t'_2 \mid \mu'$ if and only if $\varepsilon_2 t_2 :: G_1 \mid \mu \longmapsto \varepsilon'_2 t'_2 :: G_1 \mid \mu'$.*

Proof. Analogous to Lemma 92. □

Lemma 94. *$\varepsilon_1 t_1 :=^{G_3} \varepsilon_2 t_2 \mid \mu \longmapsto \varepsilon'_1 t'_1 :=^{G_3} \varepsilon_2 t_2 \mid \mu'$ if and only if $\varepsilon_1 t_1 :: \text{Ref } G_3 \mid \mu \longmapsto \varepsilon'_1 t'_1 :: \text{Ref } G_3 \mid \mu'$.*

Proof. Similar to Lemma 92. □

Lemma 95. *$\varepsilon_1 t_1 :=^{G_3} \varepsilon_2 t_2 \mid \mu \longmapsto \varepsilon_1 t_1 :=^{G_3} \varepsilon'_2 t'_2 \mid \mu'$ if and only if $\varepsilon_2 t_2 :: G_3 \mid \mu \longmapsto \varepsilon'_2 t'_2 :: G_3 \mid \mu'$.*

Proof. Analogous to Lemma 94. □

Lemma 96. *$\text{ref}^G\ \varepsilon t \mid \mu \longmapsto \text{ref}^G\ \varepsilon' t' \mid \mu'$ if and only if $\varepsilon t :: G \mid \mu \longmapsto \varepsilon' t' :: G \mid \mu'$.*

Proof. Similar to Lemma 92. □

Lemma 97. *$!^G \varepsilon t \mid \mu \longmapsto\ !^G \varepsilon' t' \mid \mu'$ if and only if $\varepsilon t :: \text{Ref } G \mid \mu \longmapsto \varepsilon' t' :: \text{Ref } G \mid \mu'$.*

Proof. Similar to Lemma 92. □

Lemma 98. *$\text{if } \varepsilon_1 t_1 \text{ then } \varepsilon_2 t_2 \text{ else } \varepsilon_3 t_3 \mid \mu \longmapsto \text{if } \varepsilon'_1 t'_1 \text{ then } \varepsilon_2 t_2 \text{ else } \varepsilon_3 t_3 \mid \mu'$ if and only if $\varepsilon_1 t_1 :: \text{Bool} \mid \mu \longmapsto \varepsilon'_1 t'_1 :: \text{Bool} \mid \mu'$.*

Proof. Similar to Lemma 92. □

Lemma 99. *If $\mathbf{c_1} \to \mathbf{c_2} = (\!| \varepsilon \vdash G'_1 \to G'_2 \sim G_1 \to G_2 |\!)$, then $\mathbf{c_1} = (\!| idom(\varepsilon) \vdash G_1 \sim G'_1 |\!)$ and $\mathbf{c_2} = (\!| icod(\varepsilon) |\!) \vdash G'_2 \sim G_2$.*

Proof. By definition of the map function between evidence augmented consistent judgments and coercions we know that $(\!| \varepsilon \vdash G_{11} \to G_{12} \sim G_{21} \to G_{22} |\!) = (\!| idom(\varepsilon) \vdash G_1 \sim G'_1 |\!) \to (\!| icod(\varepsilon) \vdash G'_2 \sim G_2 |\!)$ which is equal to $\mathbf{c_1} \to \mathbf{c_2}$, and the result holds immediately. □

Lemma 100. *If* $\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2} = (\!|\varepsilon \vdash \mathsf{Ref}\ G' \sim \mathsf{Ref}\ G|\!)$*, then* $\mathbf{c_1} = (\!|iref(\varepsilon) \vdash \mathsf{Ref}\ G \sim \mathsf{Ref}\ G'|\!)$ *and* $\mathbf{c_2} = (\!|iref(\varepsilon) \vdash \mathsf{Ref}\ G' \sim \mathsf{Ref}\ G|\!)$.

Proof. By definition of the map function between evidence augmented consistent judgments and coercions we know that $(\!|\varepsilon \vdash \mathsf{Ref}\ G' \sim \mathsf{Ref}\ G|\!) = \mathsf{Ref}\ (\!|iref(\varepsilon) \vdash G \sim G'|\!) \to (\!|iref(\varepsilon) \vdash G' \sim G|\!)$ which is equal to $\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2}$, and the result holds immediately. □

Lemma 101. *If* $t_1 \approx \mathbf{t_2}$*,* $v_1 \in \mathbb{T}[G]$*, and* $v_1 \approx \mathbf{v_2}$*, then* $t_1[v_1/x^G] \approx \mathbf{t_2}[\mathbf{v_2}/x]$

Proof. By induction on the derivation $t_1 \approx \mathbf{t_2}$. □

Lemma 102. *Consider* $\langle G\rangle \vdash G \sim G$*, then*

1. $\forall \varepsilon \vdash G \sim G'$*,* $\langle G\rangle \circ \varepsilon = \varepsilon$*, and*
2. $\forall \varepsilon \vdash G!' \sim G$*,* $\varepsilon \circ \langle G\rangle = \varepsilon$

Proof. By induction on evidence augmented consistent judgment $\langle G\rangle \vdash G \sim G$. □

Lemma 103 *(Weak bisimulation between* $\lambda^{\varepsilon}_{\widetilde{REF}}$ *and HCC*$^+$ *). If* $t_1 \in \mathbb{T}[G]$*,* $\cdot;\Sigma \vdash_H \mathbf{t_2} : G$*,* $\mu_2 \models \Sigma$*,* $\mu_1 \approx \mu_2$*, and* $t_1 \approx \mathbf{t_2}$*, then*

1. *If* $t_1 \mid \mu_1 \longmapsto t_1' \mid \mu_1'$*, then* $\mathbf{t_2} \mid \mu_2 \longmapsto^* \mathbf{t_2'} \mid \mu_2'$ *such that* $t_1' \approx \mathbf{t_2'}$ *and* $\mu_1' \approx \mu_2'$.
2. *If* $\mathbf{t_2} \mid \mu_2 \longmapsto \mathbf{t_2''} \mid \mu_2''$*, then* $\exists j, 0 \le j \le 2$*,* $\mathbf{t_2''} \mid \mu_2'' \longmapsto^j \mathbf{t_2'} \mid \mu_2'$ *and* $t_1 \mid \mu_1 \longmapsto^* t_1' \mid \mu_1'$ *such that* $t_1' \approx \mathbf{t_2'}$ *and* $\mu_1' \approx \mu_2'$.

Proof. 1. We proceed by induction on $t_1 \mid \mu_1 \longmapsto t_1' \mid \mu_1'$.

Case $(\varepsilon_1(\lambda x^{G_{11}}.t_1')\ @^{G_1 \to G_2}\ \varepsilon_2 u_1 \mid \mu_1 \longmapsto \varepsilon_1' t_1'[\varepsilon_2' u_1 :: G_{11}/x^{G_{11}}] :: G_2 \mid \mu_1)$. Where $\varepsilon_1' = icod(\varepsilon_1)$ and $\varepsilon_2' = \varepsilon_2 \circ idom(\varepsilon_1)$. By inspection of (bapp), we know that $\mathbf{t_2} = \mathbf{t_{21}}\ \mathbf{t_{22}}$, for some $\mathbf{t_{21}}, \mathbf{t_{22}}$ such that $\varepsilon_1(\lambda x^{G_{11}}.t_1') :: G_1 \to G_2 \approx \mathbf{t_{21}}$ and $\varepsilon_2 u_1 :: G_1 \approx \mathbf{t_{22}}$. We proceed by case analysis on $\mathbf{t_{21}}\ \mathbf{t_{22}}$.

- If $\mathbf{t_{21}}\ \mathbf{t_{22}} = ((\mathbf{c_1} \to \mathbf{c_2})\,(\lambda x : G_{11}.\mathbf{t_2'}))\mathbf{v_{22}}$. Where $\mathbf{c_1} \to \mathbf{c_2} = (\!|\varepsilon_1 \vdash G_{11} \to G_{12} \sim G_1 \to G_2|\!)$. By Lemma 99, $\mathbf{c_1} = (\!|idom(\varepsilon_1) \vdash G_1 \sim G_{11}|\!)$, and $\mathbf{c_2} = (\!|icod(\varepsilon_1) \vdash G_{12} \sim G_2|\!)$.
  - If $\mathbf{v_{22}} = \mathbf{c_v}\mathbf{u_{22}}$, where $\mathbf{c_v} = (\!|\varepsilon_2 \vdash G_u \sim G_1|\!)$ and $u_1 \approx \mathbf{u_{22}}$, then by Proposition 79, $\mathbf{c_1'} = \mathbf{c_v}; \mathbf{c_1} = (\!|\varepsilon_2' \vdash G_u \sim G_{11}|\!)$. Then if we assume $\mathbf{c_1'} \neq \iota_{\mathbf{G_u}}$ (the other case is analogous)

$$\begin{aligned}((\mathbf{c_1} \to \mathbf{c_2})\,(\lambda x : G_{11}.\mathbf{t_2'}))\mathbf{c_v}\mathbf{u_{22}} \mid \mu_2 &\longmapsto \mathbf{c_1}((\lambda x : G_{11}.\mathbf{t_2'})\mathbf{c_1}(\mathbf{c_v}\mathbf{u_{22}})) \mid \mu_2\\ &\longmapsto \mathbf{c_1}((\lambda x : G_{11}.\mathbf{t_2'})\mathbf{c_1'}\mathbf{u_{22}}) \mid \mu_2\\ &\longmapsto \mathbf{c_1}(\mathbf{t_2'}[\mathbf{c_1'}\mathbf{u_{22}}/x]) \mid \mu_2\end{aligned}$$

  But we know that $t_1' \approx \mathbf{t_2'}$, and that by (b::eq) $\varepsilon_2' u_1 :: G_{11} \approx \mathbf{c_1'}\mathbf{u_{22}}$, by Lemma 101 and (b::eq), $idom(\varepsilon_1)t_1'[\varepsilon_2' u_1 :: G_{11}/x^{G_{11}}] :: G_2 \approx \mathbf{c_1}(\mathbf{t_2'}[\mathbf{c_1'}\mathbf{u_{22}}/x])$ and the result holds.
  - If $\mathbf{v_{22}} = \mathbf{u_{22}}$ where $u_1 \approx \mathbf{u_{22}}$, then by (b::id) $\varepsilon_2 = \langle G_1\rangle$ and $u_1 \in \mathbb{T}[G_1]$, therefore $\langle G_1\rangle \vdash G_1 \sim G_1$. Therefore by Lemma 102, $\varepsilon_2' = idom(\varepsilon_1)$. Then

$$\begin{aligned}((\mathbf{c_1} \to \mathbf{c_2})\,(\lambda x : G_{11}.\mathbf{t_2'}))\mathbf{u_{22}} \mid \mu_2 &\longmapsto \mathbf{c_1}((\lambda x : G_{11}.\mathbf{t_2'})\mathbf{c_1}(\mathbf{u_{22}})) \mid \mu_2\\ &\longmapsto \mathbf{c_1}(\mathbf{t_2'}[\mathbf{c_1}\mathbf{u_{22}}/x]) \mid \mu_2\end{aligned}$$

  But we know that $t_1' \approx \mathbf{t_2'}$, and that by (b::eq) $\varepsilon_2' u_1 :: G_{11} \approx \mathbf{c_1}\mathbf{u_{22}}$, by Lemma 101 and (b::eq), $idom(\varepsilon_1)t_1'[\varepsilon_2' u_1 :: G_{11}/x^{G_{11}}] :: G_2 \approx \mathbf{c_1}(\mathbf{t_2'}[\mathbf{c_1}\mathbf{u_{22}}/x])$ and the result holds.
- If $\mathbf{t_{21}}\ \mathbf{t_{22}} = (\lambda x : G_1.\mathbf{t_2'})\mathbf{v_{22}}$. Then $G_{11} = G_1$, $t_1' \in \mathbb{T}[G_2]$, $\varepsilon_1 = \langle G_1 \to G_2\rangle$, and $\langle G_1 \to G_2\rangle \vdash G_1 \to G_2 \sim G_1 \to G_2$. By the inversion lemmas $idom(\varepsilon_1) = \langle G_1\rangle \vdash G_1 \sim G_1$ and $icod(\varepsilon_1) = \langle G_2\rangle \vdash G_2 \sim G_2$. But we know that $t_1' \approx \mathbf{t_2'}$, therefore by Lemma 102, $\varepsilon_2' = \varepsilon_2$. Finally by (b::id) and as $\varepsilon_2 u_1 :: G_1 \approx \mathbf{v_{22}}$, by Lemma 101, $idom(\varepsilon_1)t_1'[\varepsilon_2 u_1 :: G_{11}/x^{G_{11}}] :: G_2 \approx (\mathbf{t_2'}[\mathbf{v_{22}}/x])$ and the result holds.

Case $(\mathsf{ref}^{G_1}\ \varepsilon u_1 \mid \mu_1 \longmapsto o^{G_1} \mid \mu_1[o^{G_1} \mapsto \varepsilon u_1 :: G_1])$. We know by (b::ref) that $\mathbf{t_2} = \mathsf{ref}\ \mathbf{v_2}$, for some $\mathbf{v_2}$ such that $\varepsilon u_1 :: G_1 \approx \mathbf{v_2}$. But $\mathsf{ref}\ \mathbf{v_2} \mid \mu_2 \longmapsto o \mid \mu_2[o \mapsto \mathbf{v_2}]$. As $o^{G_1} \approx o$, and $\mu_1 \approx \mu_2$, we only have to prove that $\varepsilon u_1 :: G_1 \approx \mathbf{v_2}$, but we already know that by (b::ref), and the result holds immediately.

Case $(!^G(\varepsilon o^{G_2}) \mid \mu_1 \longmapsto iref(\varepsilon)v_1 :: G \mid \mu_1)$. Where $\mu_1(x^{G_2}) = v_1$. By inspection of (b!), we know that $\mathbf{t_2} = !\mathbf{v_2}$, for some $\mathbf{v_2}$ such that $\varepsilon o^{G_2} :: \mathsf{Ref}\ G \approx \mathbf{v_2}$. We proceed by case analysis on $\mathbf{v_2}$.

- If $\mathbf{v_2} = (\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2})o$. Where $\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2} = (\!|\varepsilon \vdash \mathsf{Ref}\ G_2 \sim \mathsf{Ref}\ G|\!)$, and $o^{G_2} \approx o$. By Lemma 100, $\mathbf{c_1} = (\!|iref(\varepsilon) \vdash G \sim G_2|\!)$, and $\mathbf{c_2} = (\!|iref(\varepsilon) \vdash G_2 \sim G|\!)$. Then

$$\begin{aligned} !(\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2})o \mid \mu_2 &\longmapsto \mathbf{c_2}!o \mid \mu_2 \\ &\longmapsto \mathbf{c_2}\mathbf{v}_2' \mid \mu_2 \end{aligned}$$

  where $\mu_2(o) = \mathbf{v}_2'$, and $\mathbf{v}_2' \approx v_1$. The result follows by applying (b::eq).
- If $\mathbf{v_2} = o$. Then $G_2 = G$ and $\varepsilon = \langle \mathsf{Ref}\ G\rangle$. By the inversion lemma on evidence $iref(\varepsilon) = \langle G\rangle \vdash G \sim G$. Then $!o \mid \mu_2 \longmapsto \mathbf{v}_2' \mid \mu_2$, where $\mu_2(o) = \mathbf{v}_2'$, and $\mathbf{v}_2' \approx v_1$. By (b::id) we know that $\langle G\rangle v_1 :: G \approx \mathbf{v}_2'$ and the result holds.

Case *($\varepsilon_1 o^{G_1} :=^{G_3} \varepsilon_2 u_{12} \mid \mu_1 \longmapsto \mathsf{unit} \mid \mu_1'$)*. Where $\mu_1' = \mu_1[o^{G_1} \mapsto \varepsilon_2' u_{12} :: G_1]$, and $\varepsilon_2' = \varepsilon_2 \circ iref(\varepsilon_1)$. By inspection of (b:=), we know that $\mathbf{t_2} = \mathbf{t_{21}}{:=}\mathbf{t_{22}}$, for some $\mathbf{t_{21}}, \mathbf{t_{22}}$ such that $\varepsilon_1 o^{G_1} :: \mathsf{Ref}\ G_3 \approx \mathbf{t_{21}}$ and $\varepsilon_2 u_{12} :: G_3 \approx \mathbf{t_{22}}$. We proceed by case analysis on $\mathbf{t_{21}}{:=}\mathbf{t_{22}}$.

- If $\mathbf{t_{21}}{:=}\mathbf{t_{22}} = ((\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2})o)\mathbf{v_{22}}$. Where $\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2} = (\!|\varepsilon_1 \vdash \mathsf{Ref}\ G_1 \sim \mathsf{Ref}\ G_3|\!)$. By Lemma 100, $\mathbf{c_1} = (\!|iref(\varepsilon_1) \vdash G_3 \sim G_1|\!)$, and $\mathbf{c_2} = (\!|iref(\varepsilon_1) \vdash G_1 \sim G_3|\!)$.
  - If $\mathbf{v_{22}} = \mathbf{c_v}\mathbf{u_{22}}$, where $\mathbf{c_v} = (\!|\varepsilon_2 \vdash G_u \sim G_3|\!)$ and $u_1 \approx \mathbf{u_{22}}$, then by Proposition 79, $\mathbf{c}_1' = \mathbf{c_v}; \mathbf{c_1} = (\!|\varepsilon_2' \vdash G_u \sim G_1|\!)$. Then if we assume $\mathbf{c}_1' \neq \iota_{\mathbf{G_u}}$ (the other case is analogous)

$$\begin{aligned} ((\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2})\,o){:=}\mathbf{c_v}\mathbf{u_{22}} \mid \mu_2 &\longmapsto o{:=}\mathbf{c_1}(\mathbf{c_v}\mathbf{u_{22}}) \mid \mu_2 \\ &\longmapsto o{:=}\mathbf{c}_1'\mathbf{u_{22}} \mid \mu_2 \\ &\longmapsto \mathsf{unit} \mid \mu_2[o \mapsto \mathbf{c}_1'\mathbf{u_{22}}] \end{aligned}$$

    But we know that $\mathsf{unit} \approx \mathsf{unit}$, and that by (b::eq) $\varepsilon_2' u_1 :: G_1 \approx \mathbf{c}_1'\mathbf{u_{22}}$, and so $\mu_1[o^{G_1} \mapsto \varepsilon_2' u_{12} :: G_1] \approx \mu_2[o \mapsto \mathbf{c}_1'\mathbf{u_{22}}]$, and the result holds.
  - If $\mathbf{v_{22}} = \mathbf{u_{22}}$ where $u_1 \approx \mathbf{u_{22}}$, then by (b::id) $\varepsilon_2 = \langle G_3\rangle$ and $u_1 \in \mathbb{T}[G_3]$, therefore $\langle G_3\rangle \vdash G_3 \sim G_3$. Therefore by Lemma 102, $\varepsilon_2' = iref(\varepsilon_1)$. Then

$$\begin{aligned} ((\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2})\,o){:=}\mathbf{u_{22}} \mid \mu_2 &\longmapsto o{:=}\mathbf{c_1}(\mathbf{u_{22}}) \mid \mu_2 \\ &\longmapsto \mathsf{unit} \mid \mu_2[o \mapsto \mathbf{c_1}\mathbf{u_{22}}] \end{aligned}$$

    But we know that $\mathsf{unit} \approx \mathsf{unit}$, and that by (b::eq) $\varepsilon_2' u_1 :: G_1 \approx \mathbf{c_1}\mathbf{u_{22}}$, and so $\mu_1[o^{G_1} \mapsto \varepsilon_2' u_{12} :: G_1] \approx \mu_2[o \mapsto \mathbf{c_1}\mathbf{u_{22}}]$, and the result holds.
- If $\mathbf{t_{21}}{:=}\mathbf{t_{22}} = o{:=}\mathbf{v_{22}}$. Then $G_1 = G_3$, $\varepsilon_1 = \langle \mathsf{Ref}\ G_3\rangle$, and $\langle G_3\rangle \vdash \mathsf{Ref}\ G_3 \sim \mathsf{Ref}\ G_3$. By the inversion lemmas $iref(\varepsilon_1) = \langle G_3\rangle \vdash G_3 \sim G_3$ and $iref(\varepsilon_1) = \langle G_3\rangle \vdash G_3 \sim G_3$. We also know that $o{:=}\mathbf{v_{22}} \mid \mu_2 \longmapsto o{:=}\mathbf{v_{22}} \mid \mu_2[o \mapsto \mathbf{v_{22}}]$, and $\mathsf{unit} \approx \mathsf{unit}$. Also by Lemma 102, $\varepsilon_2' = \varepsilon_2$. Finally by premise $\varepsilon_2 u_1 :: G_1 \approx \mathbf{v_{22}}$, and so $\mu_1[o^{G_1} \mapsto \varepsilon_2 u_{12} :: G_1] \approx \mu_2[o \mapsto \mathbf{v_{22}}]$, and the result holds.

Case *($\mathsf{if}\ \varepsilon_1 b\ \mathsf{then}\ \varepsilon_2 t_{12}\ \mathsf{else}\ \varepsilon_3 t_{13} \mid \mu_1 \longrightarrow \varepsilon_2 t_{12} :: G \mid \mu_1$)*. Where $b = \mathsf{true}$ and $\varepsilon_1 = \langle \mathsf{Bool}\rangle$. By inspection of (b), we know that $\mathbf{t_2} = \mathsf{if}\ \mathbf{t_{21}}\ \mathsf{then}\ \mathbf{t_{22}}\ \mathsf{else}\ \mathbf{t_{23}}$, for some $\mathbf{t_{21}}, \mathbf{t_{22}}, \mathbf{t_{23}}$, such that $\langle \mathsf{Bool}\rangle b :: \mathsf{Bool} \approx \mathbf{t_{21}}$, $\varepsilon_2 t_{12} :: G \approx \mathbf{t_{22}}$, and $\varepsilon_3 t_{13} :: G \approx \mathbf{t_{23}}$. By (b::leq) or (b::id) and (bb), we know that either $\mathbf{t_{21}} = \mathsf{true}$ or $\mathbf{t_{21}} = \iota_{\mathsf{Bool}}\mathsf{true}$. Let us assume $\mathbf{t_{21}} = \mathsf{true}$ (the other case is analogous modulo one extra step of evaluation). Then $\mathbf{t_2} \mid \mu_2 \longmapsto \mathbf{t_{22}} \mid \mu_2$, but $\varepsilon_2 t_{12} :: G \approx \mathbf{t_{22}}$ and the result holds immediately.

Case *($\mathsf{if}\ \varepsilon_1 b\ \mathsf{then}\ \varepsilon_2 t_{12}\ \mathsf{else}\ \varepsilon_3 t_{13} \mid \mu_1 \longrightarrow \varepsilon_2 t_{13} :: G \mid \mu_1$)*. Where $b = \mathsf{false}$ and $\varepsilon_1 = \langle \mathsf{Bool}\rangle$. By inspection of (b), we know that $\mathbf{t_2} = \mathsf{if}\ \mathbf{t_{21}}\ \mathsf{then}\ \mathbf{t_{22}}\ \mathsf{else}\ \mathbf{t_{23}}$, for some $\mathbf{t_{21}}, \mathbf{t_{22}}, \mathbf{t_{23}}$, such that $\langle \mathsf{Bool}\rangle b :: \mathsf{Bool} \approx \mathbf{t_{21}}$, $\varepsilon_2 t_{12} :: G \approx \mathbf{t_{22}}$, and $\varepsilon_3 t_{13} :: G \approx \mathbf{t_{23}}$. By (b::leq) or (b::id) and (bb), we know that either $\mathbf{t_{21}} = \mathsf{false}$ or $\mathbf{t_{21}} = \iota_{\mathsf{Bool}}\mathsf{false}$. Let us assume $\mathbf{t_{21}} = \mathsf{false}$ (the other case is analogous modulo one extra step of evaluation). Then $\mathbf{t_2} \mid \mu_2 \longmapsto \mathbf{t_{23}} \mid \mu_2$, but $\varepsilon_2 t_{13} :: G \approx \mathbf{t_{23}}$ and the result holds immediately.

Case *($\langle B_1\rangle b_1 \oplus \langle B_2\rangle b_2 \mid \mu_1 \longrightarrow b_{13} \mid \mu_1$)*. Where $b_3 = b_1 [\![\oplus]\!] b_2$. Then either $\mathbf{t_2} = \mathbf{t_{21}} \oplus \mathbf{t_{22}}$. Where by (b::leq) or (b::id) and (bb) $\mathbf{t_{21}} = c[1]$ or $\mathbf{t_{21}} = \iota_{\mathbf{B_1}} b_1$, and $\mathbf{t_{22}} = b_2$ or $\mathbf{t_{22}} = \iota_{\mathbf{B_2}} b_2$. Let us assume $\mathbf{t_{21}} = b_1$ and $\mathbf{t_{22}} = b_2$ (the other cases is analogous modulo one or two extra steps of evaluation). Then $b_1 \oplus b_2 = b_3$, where $b_3 = b_1 [\![\oplus]\!] b_2$, and the result holds immediately by (b::b).

Case *($\varepsilon_1 t_{11} \ @^{G_1 \to G_2}\ \varepsilon_2 t_{12} \mid \mu_1 \longmapsto \varepsilon_1' t_{11}' \ @^{G_1 \to G_2}\ \varepsilon_2 t_{12} \mid \mu_1'$)*. Then by (bapp) $\mathbf{t_2} = \mathbf{t_{21}}\ \mathbf{t_{22}}$. By Lemma 92 we know that $\varepsilon_1 t_{11} :: G_1 \to G_2 \mid \mu_1 \longmapsto \varepsilon_1' t_{11}' :: G_1 \to G_2 \mid \mu_1'$. Also by (bapp) we know that $\varepsilon_1 t_{11} :: G_1 \to G_2 \approx \mathbf{t_{21}}$. Then by induction hypothesis we know that $\mathbf{t_{21}} \mid \mu_2 \longmapsto^* \mathbf{t}_{21}' \mid \mu_2'$, and that $\varepsilon_1' t_{11}' :: G_1 \to G_2 \approx \mathbf{t}_{21}'$ and $\mu_1' \approx \mu_2'$. The result follows directly by (bapp).

Case *($\varepsilon_1 u_{11} \mathbin{@^{G_1\to G_2}} \varepsilon_2 t_{12} \mid \mu_1 \longmapsto \varepsilon_1 u_{11} \mathbin{@^{G_1\to G_2}} \varepsilon_2' t_{12}' \mid \mu_1'$).* Analogous to previous case but using Lemma 93

Case *($\text{if } \varepsilon_1 t_{11} \text{ then } \varepsilon_2 t_{12} \text{ else } \varepsilon_3 t_{13} \mid \mu_1 \longmapsto \text{if } \varepsilon_1 t_{11}' \text{ then } \varepsilon_2 t_{12} \text{ else } \varepsilon_3 t_{13} \mid \mu_1'$).* Then by (bif) $\mathbf{t_2} = \text{if } \mathbf{t_{21}} \text{ then } \mathbf{t_{22}} \text{ else } \mathbf{t_{23}}$. By Lemma 98 we know that $\varepsilon_1 t_{11} :: \text{Bool} \mid \mu_1 \longmapsto \varepsilon_1' t_{11}' :: \text{Bool} \mid \mu_1'$. Also by (bif) we know that $\varepsilon_1 t_{11} :: \text{Bool} \approx \mathbf{t_{21}}$. Then by induction hypothesis we know that $\mathbf{t_{21}} \mid \mu_2 \longmapsto^* \mathbf{t_{21}'} \mid \mu_2'$, and that $\varepsilon_1' t_{11}' :: \text{Bool} \approx \mathbf{t_{21}'}$ and $\mu_1' \approx \mu_2'$. The result follows directly by (bif).

Case *($\varepsilon_1 t_{11} :=^{G_3} \varepsilon_2 t_{12} \mid \mu_1 \longmapsto \varepsilon_1' t_{11}' :=^{G_3} \varepsilon_2 t_{12} \mid \mu_1'$).* Then by (b:=) $\mathbf{t_2} = \mathbf{t_{21}} {:=} \mathbf{t_{22}}$. By Lemma 94 we know that $\varepsilon_1 t_{11} :: \text{Ref } G_3 \mid \mu_1 \longmapsto \varepsilon_1' t_{11}' :: \text{Ref } G_3 \mid \mu_1'$. Also by (b:=) we know that $\varepsilon_1 t_{11} :: \text{Ref } G_3 \approx \mathbf{t_{21}}$. Then by induction hypothesis we know that $\mathbf{t_{21}} \mid \mu_2 \longmapsto^* \mathbf{t_{21}'} \mid \mu_2'$, and that $\varepsilon_1' t_{11}' :: \text{Ref } G_3 \approx \mathbf{t_{21}'}$ and $\mu_1' \approx \mu_2'$. The result follows directly by (b:=).

Case *($\varepsilon_1 u_{11} :=^{G_3} \varepsilon_2 t_{12} \mid \mu_1 \longmapsto \varepsilon_1 u_{11} :=^{G_3} \varepsilon_2' t_{12}' \mid \mu_1'$).* Analogous to previous case but using Lemma 95

Case *($\text{ref}^{G'} \varepsilon_1 t_{11} \mid \mu_1 \longmapsto \text{ref}^{G'} \varepsilon_1' t_{11}' \mid \mu_1'$).* Then by (bref) $\mathbf{t_2} = \text{ref } \mathbf{t_{21}}$. By Lemma 96 we know that $\varepsilon_1 t_{11} :: G' \mid \mu_1 \longmapsto \varepsilon_1' t'[11] :: G' \mid \mu_1'$. Also by (bref) we know that $\varepsilon_1 t_{11} :: G' \approx \mathbf{t_{21}}$. Then by induction hypothesis we know that $\mathbf{t_{21}} \mid \mu_2 \longmapsto^* \mathbf{t_{21}'} \mid \mu_2'$, and that $\varepsilon_1' t_{11}' :: G' \approx \mathbf{t_{21}'}$ and $\mu_1' \approx \mu_2'$. The result follows directly by (bref).

Case *($!^{G'} \varepsilon_1 t_{11} \mid \mu_1 \longmapsto !^{G'} \varepsilon_1' t_{11}' \mid \mu_1'$).* Then by (b!) $\mathbf{t_2} = !\mathbf{t_{21}}$. By Lemma 97 we know that $\varepsilon_1 t_{11} :: \text{Ref } G' \mid \mu_1 \longmapsto \varepsilon_1' t_{11}' :: \text{Ref } G' \mid \mu_1'$. Also by (b!) we know that $\varepsilon_1 t_{11} :: \text{Ref } G' \approx \mathbf{t_{21}}$. Then by induction hypothesis we know that $\mathbf{t_{21}} \mid \mu_2 \longmapsto^* \mathbf{t_{21}'} \mid \mu_2'$, and that $\varepsilon_1' t_{11}' :: \text{Ref } G' \approx \mathbf{t_{21}'}$ and $\mu_1' \approx \mu_2'$. The result follows directly by (b!).

Case *($\varepsilon_1 t_{11} :: G \mid \mu_1 \longmapsto \varepsilon_1' t_1' :: G \mid \mu_1'$).* Without loosing generality let us assume that $\varepsilon_1 t_{11} :: G = \varepsilon_1(\ldots(\varepsilon_n t_{1n} :: G_n)\ldots) :: G$, where $t_{1n} \in \mathbb{T}[G_{n-1}]$ is not an ascribed term.

Then by (b::eq), (b::id) and (b::leq), either $\mathbf{t_2} = \mathbf{c_1}(\ldots(\mathbf{c_m t_{2m}})\ldots)$ (and $m \leq n$) or $\mathbf{t_2} = \mathbf{t_{21}}$, where $\mathbf{t_{2m}}$ and $\mathbf{t_{21}}$ are not coerced terms, such that $t_{1n} \approx \mathbf{t_{2m}}$ or $t_{1n} \approx \mathbf{t_{21}}$ respectively.

Let us assume that $\mathbf{t_2} = \mathbf{c_1}(\ldots(\mathbf{c_m t_{2m}})\ldots)$ (the other case is analogous to the second subcase below). Then we know that $\mathbf{c_1}(\ldots(\mathbf{c_m t_{2m}})\ldots) \longmapsto \mathbf{c_1' t_{2m}}$, where $\mathbf{c_m}; \mathbf{c_{m-1}} \ldots; \mathbf{c_1} \longmapsto^* \mathbf{c_1'}$ and nm $\mathbf{c_1'}$. Also by repeatedly applying (b::eq) and (b::leq), $\varepsilon_1(\ldots(\varepsilon_n t_{1n} :: G_n)\ldots) :: G \approx \mathbf{c_1' t_{2m}}$. Additionally, by inspection of (b::eq) and (b::leq), $\exists \mathbf{c_{11}'}, \ldots \mathbf{c_{1n}'}$, such that $\mathbf{c_{1i}'} = (\!|\varepsilon_i \vdash G_{i+1} \sim G_i|\!)$, and that $\mathbf{c_{1n}'}; \ldots \mathbf{c_{11}'} \longmapsto^* \mathbf{c_1'}$.

We now proceed by case analysis on t_{1n}.

- If $t_{1n} = u_1$. Then $\varepsilon_1(\ldots(\varepsilon_{n-1}(\varepsilon_n u_1 :: G_n) :: G_{n-1})\ldots) :: G \longmapsto \varepsilon_1(\ldots(\varepsilon_{n-1}' u_1 :: G_{n-1})\ldots) :: G$, where $\varepsilon_{n-1}' = (\varepsilon_n \circ \varepsilon_{n-1})$. Then $\mathbf{c_{1n}'}; \mathbf{c_{1n-1}'} \longmapsto^* \mathbf{c_{1n-1}''}$, for some $\mathbf{c_{1n-1}''}$, therefore by Lemma 79, $\mathbf{c_{1n-1}''} = (\!|\varepsilon_{n-1}' \vdash G_{n+1} \sim G_{n-1}|\!)$. Then by (b::eq), $\varepsilon_{n-1}' u_1 :: G_{n-1} \approx \mathbf{c_{1n-1}'' u_2}$. Then the result holds by using (b::leq) and (b::eq) repeatedly and using $\mathbf{c_{11}'}, \ldots, \mathbf{c_{1n-2}'}, \mathbf{c_{1n-1}''}$ and Lemma 90.
- If t_{1n} is not a simple value, and therefore $t_{1n} \mid \mu_1 \longmapsto t_{1n}' \mid \mu_1'$. By induction hypothesis $\mathbf{t_{2m}} \mid \mu_2 \longmapsto^* \mathbf{t_{2m}'} \mid \mu_2'$, such that $t_{1n}' \approx \mathbf{t_{2m}'}$ and $\mu_1' \approx \mu_2'$ Therefore by (b::leq) and (b::eq), $\varepsilon_1(\ldots(\varepsilon_n t_{1n}' :: G_n)\ldots) :: G \approx \mathbf{c_1' t_{2m}'}$ and the result holds.

The proof of (2) is similar but choosing sometimes $j = 1$ or $j = 2$ in cases for application, dereference or assignment. □

Lemma 104. *If $\mathbf{c} = (\!|\varepsilon \vdash G' \sim G|\!)$, then* nm $\mathbf{c}$.

Proof. Direct by induction on $\varepsilon \vdash G' \sim G$ and definition of $(\!|\,|\!)$ (Fig. 12). □

Lemma 105. *If $t_1 \in \mathbb{T}[G]$, $\varnothing; \Sigma \vdash_H \mathbf{t_2} : G$, $\mu_2 \models \Sigma$, $\mu_1 \approx \mu_2$, and $t_1 \approx \mathbf{t_2}$, then $t_1 \mid \mu_1 \Downarrow \iff \mathbf{t_2} \mid \mu_2 \Downarrow$.*

Proof.

Case *($\Rightarrow$).* We proceed by induction on $t_1 \mid \mu_1 \longmapsto^* v_1 \mid \mu_1'$.

Case *($t_1 = v_1$).* As $t_1 \approx \mathbf{t_2}$ and by Lemma 15, then $\mathbf{t_2} \mid \mu_2 \longmapsto^* \mathbf{t_2'} \mid \mu_2'$, and $v_1 \approx \mathbf{t_2'}$ and $\mu_1 \approx \mu_2'$. If $v_1 = u$, then the result holds immediately by inspection of (B*b*), (bλ), and (b*o*). If $v_1 = \varepsilon u :: G$ then either by (b::id) $\mathbf{t_2'} = \mathbf{u_2}$ and the result holds, or by (b::eq) $\mathbf{t_2'} = \mathbf{c u_2}$, where $\mathbf{c} = (\!|\varepsilon \vdash G_u \sim G|\!)$ (and therefore $\mathbf{c} \neq \text{Fail}$) and by Lemma 104 the result holds.

Case *($t_1 \mid \mu_1 \longmapsto t_1' \mid \mu_1''$ and $t_1' \mid \mu_1'' \longmapsto^* v_1 \mid \mu_1'$).* By Lemma 15 then $\mathbf{t_2} \mid \mu_2 \longmapsto^* \mathbf{t_2'} \mid \mu_2'$ and $t_1' \approx \mathbf{t_2'}$ and $\mu_1' \approx \mu_2$. Then by induction hypothesis, $\mathbf{t_2'} \mid \mu_2'' \Downarrow$, and therefore $\mathbf{t_2} \mid \mu_2 \Downarrow$ and the result holds.

Case *($\Leftarrow$).* We proceed similarly by induction on $\mathbf{t_2} \mid \mu_2 \longmapsto^* \mathbf{v_2} \mid \mu_2'$.

Case *($\mathbf{t_2} = \mathbf{v_2}$).* Similar to the ($t_1 = v_1$) case.

Case *($\mathbf{t_2} \mid \mu_2 \longmapsto^k \mathbf{t_2'} \mid \mu_2''$ and $\mathbf{t_2'} \mid \mu_2'' \longmapsto^{n-k} \mathbf{v_2} \mid \mu_2'$).* By Lemma 15 we know that there exists some $j \in \{1,2,3\}$, such that $t_1 \mid \mu_1 \longmapsto^* t_1' \mid \mu_1''$ and $t_1' \approx \mathbf{t_2'}$ and $\mu_1' \approx \mu_2$. We choose $k = j$, and by induction hypothesis, $t_1' \mid \mu_2'' \Downarrow$, and therefore $t_1' \mid \mu_1 \Downarrow$ and the result holds. □

Lemma 106. *If $t_1 \in \mathbb{T}[G]$, $\emptyset; \Sigma \vdash_H \mathbf{t_2} : G$, $\mu_2 \models \Sigma$, $\mu_1 \approx \mu_2$, and $t_1 \approx \mathbf{t_2}$, then*

1. *If $t_1 \mid \mu_1 \longmapsto$ **error**, then $\mathbf{t_2} \mid \mu_2 \longmapsto^*$ **error**.*
2. *If $\mathbf{t_2} \mid \mu_2 \longmapsto r$, for some $r = \mathbf{t_2'} \mid \mu_2'$ or $r =$ **error**, then $\exists j, 0 \leq j \leq 2$. if $r \longmapsto^j$ **error**, then $t_1 \mid \mu_1 \longmapsto^*$ **error***

Proof. 1. We proceed by induction on $t_1 \mid \mu_1 \longmapsto$ **error**.

Case *($\varepsilon_1(\lambda x^{G_{11}}.t_1') \,@^{G_1 \to G_2}\, \varepsilon_2 u_1 \mid \mu_1 \longmapsto$ **error**).* Where $\varepsilon_2 \circ idom(\varepsilon_1)$ is not defined. By inspection of (bapp), we know that $\mathbf{t_2} = \mathbf{t_{21}}\ \mathbf{t_{22}}$, for some $\mathbf{t_{21}}, \mathbf{t_{22}}$ such that $\varepsilon_1(\lambda x^{G_{11}}.t_1') :: G_1 \to G_2 \approx \mathbf{t_{21}}$ and $\varepsilon_2 u_1 :: G_1 \approx \mathbf{t_{22}}$. We proceed by case analysis on $\mathbf{t_{21}}\ \mathbf{t_{22}}$.

- If $\mathbf{t_{21}}\ \mathbf{t_{22}} = ((\mathbf{c_1} \to \mathbf{c_2})\,(\lambda x : G_{11}.\mathbf{t_2'}))\mathbf{v_{22}}$. Where $\mathbf{c_1} \to \mathbf{c_2} = (\!|\varepsilon_1 \vdash G_{11} \to G_{12} \sim G_1 \to G_2|\!)$. By Lemma 99, $\mathbf{c_1} = (\!|idom(\varepsilon_1) \vdash G_1 \sim G_{11}|\!)$.
  - If $\mathbf{v_{22}} = \mathbf{c_v}\mathbf{u_{22}}$, where $\mathbf{c_v} = (\!|\varepsilon_2 \vdash G_u \sim G_1|\!)$ and $u_1 \approx \mathbf{u_{22}}$, then by Proposition 79, $\mathbf{c_1'} = \mathbf{c_v}; \mathbf{c_1} = \mathsf{Fail}$.

$$\begin{aligned} ((\mathbf{c_1} \to \mathbf{c_2})\,(\lambda x : G_{11}.\mathbf{t_2'}))\mathbf{c_v}\mathbf{u_{22}} \mid \mu_2 &\longmapsto \mathbf{c_1}((\lambda x : G_{11}.\mathbf{t_2'})\mathbf{c_1}(\mathbf{c_v}\mathbf{u_{22}})) \mid \mu_2 \\ &\longmapsto \mathbf{c_1}((\lambda x : G_{11}.\mathbf{t_2'})\mathbf{c_1'}\mathbf{u_{22}}) \mid \mu_2 \\ &\longmapsto \mathbf{error} \end{aligned}$$

  and the result holds.
  - If $\mathbf{v_{22}} = \mathbf{u_{22}}$. This case cannot happen: as $u_1 \approx \mathbf{u_{22}}$, then by (b::id) $\varepsilon_2 = \langle G_1 \rangle$ and $u_1 \in \mathbb{T}[G_1]$, therefore $\langle G_1 \rangle \vdash G_1 \sim G_1$. Therefore by Lemma 102, $\varepsilon_2 \circ idom(\varepsilon_1)$ is defined, which is a contradiction.
- If $\mathbf{t_{21}}\ \mathbf{t_{22}} = (\lambda x : G_1.\mathbf{t_2'})\mathbf{v_{22}}$. This case cannot happen as $\varepsilon_1 = \langle G_1 \to G_2 \rangle$ and as $\varepsilon_2 \vdash G_u \sim G_1$, $\varepsilon_2 \circ idom(\varepsilon_1) = \varepsilon_2 \circ \langle G_1 \rangle$ by Lemma 102 it would never fail.

Case *($\varepsilon_1 o^{G_1} :=^{G_3} \varepsilon_2 u_{12} \mid \mu_1 \longmapsto$ **error**).* Where $\varepsilon_2 \circ iref(\varepsilon_1)$ is not defined. By inspection of (b:=), we know that $\mathbf{t_2} = \mathbf{t_{21}} {:=} \mathbf{t_{22}}$, for some $\mathbf{t_{21}}, \mathbf{t_{22}}$ such that $\varepsilon_1 o^{G_1} :: \mathsf{Ref}\ G_3 \approx \mathbf{t_{21}}$ and $\varepsilon_2 u_{12} :: G_3 \approx \mathbf{t_{22}}$. We proceed by case analysis on $\mathbf{t_{21}} {:=} \mathbf{t_{22}}$.

- If $\mathbf{t_{21}} {:=} \mathbf{t_{22}} = ((\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2})o)\mathbf{v_{22}}$. Where $\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2} = (\!|\varepsilon_1 \vdash \mathsf{Ref}\ G_1 \sim \mathsf{Ref}\ G_3|\!)$. By Lemma 100, $\mathbf{c_1} = (\!|iref(\varepsilon_1) \vdash G_3 \sim G_1|\!)$, and $\mathbf{c_2} = (\!|iref(\varepsilon_1) \vdash G_1 \sim G_3|\!)$.
  - If $\mathbf{v_{22}} = \mathbf{c_v}\mathbf{u_{22}}$, where $\mathbf{c_v} = (\!|\varepsilon_2 \vdash G_u \sim G_3|\!)$ and $u_1 \approx \mathbf{u_{22}}$, then by Proposition 79, $\mathbf{c_1'} = \mathbf{c_v}; \mathbf{c_1} = \mathsf{Fail}$.

$$\begin{aligned} ((\mathsf{Ref}\ \mathbf{c_1}\ \mathbf{c_2})\,o){:=}\mathbf{c_v}\mathbf{u_{22}} \mid \mu_2 &\longmapsto o{:=}\mathbf{c_1}(\mathbf{c_v}\mathbf{u_{22}}) \mid \mu_2 \\ &\longmapsto o{:=}\mathbf{c_1'}\mathbf{u_{22}} \mid \mu_2 \\ &\longmapsto \mathbf{error} \end{aligned}$$

  And the result holds.
  - If $\mathbf{v_{22}} = \mathbf{u_{22}}$. This case cannot happen: as $u_1 \approx \mathbf{u_{22}}$, then by (b::id) $\varepsilon_2 = \langle G_3 \rangle$ and $u_1 \in \mathbb{T}[G_3]$, therefore $\langle G_3 \rangle \vdash G_3 \sim G_3$. Therefore by Lemma 102, $\varepsilon_2 \circ idom(\varepsilon_1)$ is defined, which is a contradiction.
- If $\mathbf{t_{21}} {:=} \mathbf{t_{22}} = o{:=}\mathbf{v_{22}}$. This case cannot happen as $\varepsilon_1 = \langle \mathsf{Ref}\ G_3 \rangle$ and as $\varepsilon_2 \vdash G_u \sim G_3$, $\varepsilon_2 \circ iref(\varepsilon_1) = \varepsilon_2 \circ \langle G_3 \rangle$ by Lemma 102 it would never fail.

Case *($\varepsilon_1 t_{11} \,@^{G_1 \to G_2}\, \varepsilon_2 t_{12} \mid \mu_1 \longmapsto$ **error**, $t_{11} \neq u$).* Then by (bapp) $\mathbf{t_2} = \mathbf{t_{21}}\ \mathbf{t_{22}}$. By Lemma 92 we know that $\varepsilon_1 t_{11} :: G_1 \to G_2 \mid \mu_1 \longmapsto$ **error**. Also by (bapp) we know that $\varepsilon_1 t_{11} :: G_1 \to G_2 \approx \mathbf{t_{21}}$. Then by induction hypothesis we know that $\mathbf{t_{21}} \mid \mu_2 \longmapsto^*$ **error**, and the result holds.

Case *($\mathsf{if}\ \varepsilon_1 t_{11}\ \mathsf{then}\ \varepsilon_2 t_{12}\ \mathsf{else}\ \varepsilon_3 t_{13} \mid \mu_1 \longmapsto$ **error**, $t_{11} \neq u$).* Then by (bif) $\mathbf{t_2} = \mathsf{if}\ \mathbf{t_{21}}\ \mathsf{then}\ \mathbf{t_{22}}\ \mathsf{else}\ \mathbf{t_{23}}$. By Lemma 98 we know that $\varepsilon_1 t_{11} :: \mathsf{Bool} \mid \mu_1 \longmapsto$ **error**. Also by (bif) we know that $\varepsilon_1 t_{11} :: \mathsf{Bool} \approx \mathbf{t_{21}}$. Then by induction hypothesis we know that $\mathbf{t_{21}} \mid \mu_2 \longmapsto^*$ **error**, and the result holds.

Case *($\varepsilon_1 u_{11} \,@^{G_1 \to G_2}\, \varepsilon_2 t_{12} \mid \mu_1 \longmapsto$ **error**, $t_{12} \neq u$).* Analogous to previous case but using Lemma 93.

Case *($\varepsilon_1 t_{11} :=^{G_3} \varepsilon_2 t_{12} \mid \mu_1 \longmapsto$ **error**$, t_{11} \neq u$).* Then by (b:=) $\mathbf{t_2} = \mathbf{t_{21}{:=}t_{22}}$. By Lemma 94 we know that $\varepsilon_1 t_{11} :: \mathsf{Ref}\ G_3 \mid \mu_1 \longmapsto$ **error**. Also by (b:=) we know that $\varepsilon_1 t_{11} :: \mathsf{Ref}\ G_3 \approx \mathbf{t_{21}}$. Then by induction hypothesis we know that $\mathbf{t_{21}} \mid \mu_2 \longmapsto^*$ **error**, and the result follows.

Case *($\varepsilon_1 u_{11} :=^{G_3} \varepsilon_2 t_{12} \mid \mu_1 \longmapsto$ **error**$, t_{12} \neq u$).* Analogous to previous case but using Lemma 95.

Case *($\mathsf{ref}^{G'}\ \varepsilon_1 t_{11} \mid \mu_1 \longmapsto$ **error**$, t_{11} \neq u$).* Then by (bref) $\mathbf{t_2} = \mathsf{ref}\ \mathbf{t_{21}}$. By Lemma 96 we know that $\varepsilon_1 t_{11} :: G' \mid \mu_1 \longmapsto$ **error**. Also by (bref) we know that $\varepsilon_1 t_{11} :: G' \approx \mathbf{t_{21}}$. Then by induction hypothesis we know that $\mathbf{t_{21}} \mid \mu_2 \longmapsto^*$ **error**, and the result follows.

Case *($!^{G'} \varepsilon_1 t_{11} \mid \mu_1 \longmapsto$ **error**$, t_{11} \neq u$).* Then by (b!) $\mathbf{t_2} = !\mathbf{t_{21}}$. By Lemma 97 we know that $\varepsilon_1 t_{11} :: \mathsf{Ref}\ G' \mid \mu_1 \longmapsto$ **error**. Also by (b!) we know that $\varepsilon_1 t_{11} :: \mathsf{Ref}\ G' \approx \mathbf{t_{21}}$. Then by induction hypothesis we know that $\mathbf{t_{21}} \mid \mu_2 \longmapsto^*$ **error**, and the result follows.

Case *($\varepsilon_1 t_{11} :: G \mid \mu_1 \longmapsto \varepsilon_1' t_1' :: G \mid \mu_1'$).* Without loosing generality let us assume that $\varepsilon_1 t_{11} :: G = \varepsilon_1(...(\varepsilon_n t_{1n} :: G_n)...) :: G$, where $t_{1n} \in \mathbb{T}[G_{n-1}]$ is not an ascribed term.

Then by (b::eq), (b::id) and (b::leq), either $\mathbf{t_2} = \mathbf{c_1}(...(\mathbf{c_m t_{2m}})...)$ (and $m \leq n$) or $\mathbf{t_2} = \mathbf{t_{21}}$, where $\mathbf{t_{2m}}$ and $\mathbf{t_{21}}$ are not coerced terms, such that $t_{1n} \approx \mathbf{t_{2m}}$ or $t_{1n} \approx \mathbf{t_{21}}$ respectively.

Let us assume that $\mathbf{t_2} = \mathbf{c_1}(...(\mathbf{c_m t_{2m}})...)$ (the other case is analogous to the second subcase below). Then we know that $\mathbf{c_1}(...(\mathbf{c_m t_{2m}})...) \longmapsto \mathbf{c_1' t_{2m}}$, where $\mathbf{c_m}; \mathbf{c_{m-1}}...; \mathbf{c_1} \longmapsto^* \mathbf{c_1'}$ and nm $\mathbf{c_1'}$. Also by repeatedly applying (b::eq) and (b::leq), $\varepsilon_1(...(\varepsilon_n t_{1n} :: G_n)...) :: G \approx \mathbf{c_1' t_{2m}}$. Additionally, by inspection of (b::eq) and (b::leq), $\exists \mathbf{c_{11}'}, ...\mathbf{c_{1n}'}$, such that $\mathbf{c_{1i}'} = (\!|\varepsilon_i \vdash G_{i+1} \sim G_i|\!)$, and that $\mathbf{c_{1n}'}; ...\mathbf{c_{11}'} \longmapsto^* \mathbf{c_1'}$.

We now proceed by case analysis on t_{1n}.

- If $t_{1n} = u_1$. Then $\mathbf{t_{2m}} = \mathbf{u_2}$ for some $\mathbf{u_2}$, also $\varepsilon_1(...(\varepsilon_{n-1}(\varepsilon_n u_1 :: G_n) :: G_{n-1})...) :: G \longmapsto$ **error**, where $(\varepsilon_n \circ \varepsilon_{n-1})$ is not defined. Then by Lemma 79, $\mathbf{c_{1n}'}; \mathbf{c_{1n-1}'} \longmapsto^* \mathsf{Fail}$, therefore by Lemma 90 $\mathbf{c_1'} = \mathsf{Fail}$, but $\mathsf{Fail}\ \mathbf{u_2} \longmapsto$ **error** and the result holds.
- If t_{1n} is not a simple value, and therefore $t_{1n} \mid \mu_1 \longmapsto$ **error**. By induction hypothesis $\mathbf{t_{2m}} \mid \mu_2 \longmapsto^*$ **error** and the result holds.

The proof of (2) is similar but choosing sometimes $j = 2$ or $j = 3$ in cases for application, dereference or assignment. □

Lemma 107. *Let $G_1 \neq G_2$ such that $G_1 \sim G_2$, then $\langle\!\langle G_1 \Rightarrow G_2 \rangle\!\rangle = (\!|\mathcal{I}(G_1, G_2) \vdash G_1 \sim G_2|\!)$*

Proof. Straightforward induction on $G_1 \sim G_2$. □

Lemma 108. *If $t_1 \in \mathbb{T}[G]$, $\varnothing; \Sigma \vdash_H \mathbf{t_2} : G$, $\mu_2 \models \Sigma$, $\mu_1 \approx \mu_2$, and $t_1 \approx \mathbf{t_2}$, then $t_1 \mid \mu_1 \Downarrow$ **error** $\iff \mathbf{t_2} \mid \mu_2 \Downarrow$ **error**.*

Proof. Similar to Lemma 105 □

Proposition 109 *(Translations are bisimilar). Given $t : G$, if $t \rightsquigarrow_n t_1 : G$, and $t \rightsquigarrow_c \mathbf{t_2} : G$, then $t_1 \approx \mathbf{t_2}$.*

Proof. We prove the proposition on open terms: If $\Gamma; \varnothing \vdash t : G$, $\Gamma; \varnothing \vdash t \rightsquigarrow_n t_1 : G$, and $\Gamma; \varnothing \vdash t \rightsquigarrow_c \mathbf{t_2} : G$, then $t_1 \approx \mathbf{t_2}$.

We proceed by induction on $\Gamma; \varnothing \vdash t : G$ (we only show some cases as the others are analogous).

Case *($\Gamma; \varnothing \vdash t' :: G : G$).* Then

$$(TR::)\frac{\Gamma; \varnothing \vdash t' \rightsquigarrow_n t^{G'} : G' \qquad \varepsilon = \mathcal{I}_=(G', G)}{\Gamma; \varnothing \vdash (t' :: G) \rightsquigarrow_n (\varepsilon t^{G'} :: G) : G} \qquad (HR::)\frac{\Gamma; \varnothing \vdash t' \rightsquigarrow_c \mathbf{t'} : G'}{\Gamma; \varnothing \vdash (t' :: G) \rightsquigarrow_c \langle G' \Rightarrow G\rangle \mathbf{t'} : G}$$

By $(G\ ::)$ we know that $\Gamma; \varnothing \vdash t' : G'$, then by induction hypothesis $t^{G'} \approx \mathbf{t'}$. If $G' = G$, then $\varepsilon = \langle G \rangle$ and $\langle G \Rightarrow G\rangle \mathbf{t'} = \mathbf{t'}$, therefore the result holds immediately by (b::id). If $G' \neq G$, then by Lemma 107, $\langle G' \Rightarrow G \rangle = \langle\!\langle G' \Rightarrow G \rangle\!\rangle = (\!|\varepsilon \vdash G' \sim G|\!)$, and the result holds immediately by (b::eq).

Case *($\Gamma; \varnothing \vdash \lambda x : G_1.t' : G_1 \to G_2$).* Then

$$(TR\lambda)\frac{\Gamma, x : G_1 \vdash t' \rightsquigarrow_n t^{G_2} : G_2}{\Gamma; \varnothing \vdash \lambda x : G_1.t' \rightsquigarrow_n \lambda x^{G_1}.t^{G_2} : G_1 \to G_2} \qquad (HR\lambda)\frac{\Gamma, x : G_1 \vdash t' \rightsquigarrow_c \mathbf{t_2'} : G_2}{\Gamma; \varnothing \vdash (\lambda x : G_1.t') \rightsquigarrow_c (\lambda x : G_1.\mathbf{t_2'}) : G_1 \to G_2}$$

By $(G\lambda)$ we know that $\Gamma, x : G_1 \vdash t' : G_2$, then by induction hypothesis $t^{G_2} \approx \mathbf{t_2'}$. Then the result holds immediately by $(\mathrm{b}\lambda)$.

Case *($\Gamma; \varnothing \vdash t_1 := t_2 : \mathsf{Unit}$).*

$$(TRasgn)\frac{\begin{array}{c}\Gamma; \varnothing \vdash t_1 \rightsquigarrow_n t^{G_1} : G_1 \qquad \Gamma; \varnothing \vdash t_2 \rightsquigarrow_n t^{G_2} : G_2 \\ G_3 = \widetilde{tref}(G_1) \quad \varepsilon_1 = \mathcal{I}_=(G_1, \mathsf{Ref}\ G_3) \quad \varepsilon_2 = \mathcal{I}_=(G_2, G_3)\end{array}}{\Gamma; \varnothing \vdash t_1 := t_2 \rightsquigarrow_n \varepsilon_1 t^{G_1} :=^{G_3} \varepsilon_2 t^{G_2} : \mathsf{Unit}}$$

$$(HRasgn)\frac{\Gamma; \varnothing \vdash t_1 \rightsquigarrow_c \mathbf{t_1'} : G_1 \quad \Gamma; \varnothing \vdash t_2 \rightsquigarrow_c \mathbf{t_2'} : G_2 \quad G_3 = \widetilde{tref}(G_1)}{\Gamma; \varnothing \vdash t_1 := t_2 \rightsquigarrow_c \langle G_1 \Rightarrow \mathsf{Ref}\ G_3\rangle \mathbf{t_1'} \mathbf{:=} \langle G_2 \Rightarrow G_3\rangle \mathbf{t_2'} : \mathsf{Unit}}$$

By $(G :=)$ $\Gamma; \varnothing \vdash t_1 : G_1$ and $\Gamma; \varnothing \vdash t_2 : G_2$, therefore by induction hypothesis $t^{G_1} \approx \mathbf{t_1'}$ and $t^{G_2} \approx \mathbf{t_2'}$. Let us consider $G_1 \neq \mathsf{Ref}\ G_3$ and $G_2 \neq G_3$ (the other cases are similar – see case for ascription). By Lemma 107, $\langle G_1 \Rightarrow \mathsf{Ref}\ G_3\rangle = \langle\!\langle G_1 \Rightarrow \mathsf{Ref}\ G_3\rangle\!\rangle = (\!|\varepsilon_1 \vdash G_1 \sim \mathsf{Ref}\ G_3|\!)$, therefore by (b::eq), $\varepsilon_1 t^{G_1} :: \mathsf{Ref}\ G_3 \approx \langle\!\langle G_1 \Rightarrow \mathsf{Ref}\ G_3\rangle\!\rangle \mathbf{t_1'}$. By using similar argument, $\varepsilon_2 t G_2 :: G_3 \approx \langle\!\langle G_2 \Rightarrow G_3\rangle\!\rangle \mathbf{t_2'}$. Then the result holds by (b:=). ☐

Corollary 110. *Given $t : G$, if $t \rightsquigarrow_n t_1 : G$ and $t \rightsquigarrow_c \mathbf{t_2} : G$, then $t_1 \Downarrow \iff \mathbf{t_2} \Downarrow$ and $t_1 \Downarrow$ **error** $\iff \mathbf{t_2} \Downarrow$ **error**. (Co-divergence follows trivially.)*

Proof. By Proposition 16, and then combining Lemmas 105 and 108. ☐

Appendix F. Encoding permissive and monotonic references in $\lambda_{\widetilde{\mathrm{REF}}}$

Lemma 111. *If $et \mid \nu \longrightarrow_c et' \mid \nu'$, then*

1. *$ev(et') \sqsubseteq ev(et)$*
2. *$\forall o^{G'}_{\mathsf{m}} \in dom(\nu), ev(\nu'(o^{G'}_{\mathsf{m}})) \sqsubseteq ev(\nu(o^{G'}_{\mathsf{m}}))$*

Proof.

Case *($et = \langle G_2\rangle(\langle G_1\rangle o^{G_5}_{\mathsf{m}} :: G''), \nu(o^{G_5}_{\mathsf{m}}) = \langle G'\rangle u' :: G_5, G_3 = G_1 \sqcap G_2$, and $G' \neq \widetilde{tref}(G_3)$).* Then $et \mid \nu \longmapsto \langle G_3\rangle o^{G_5}_{\mathsf{m}} \mid \nu[o^{G_5}_{\mathsf{m}} \mapsto \langle G' \sqcap \widetilde{tref}(G_3)\rangle \nu(o^{G_5}_{\mathsf{m}}) :: G_5]$. We have to prove that $G_1 \sqcap G_2 \sqsubseteq G_2$, and that $G' \sqcap \widetilde{tref}(G_3) \sqsubseteq G'$ which is immediate from Proposition 82.

Case *(otherwise).* Then $\langle G_2\rangle(\langle G_1\rangle u :: G'') \mid \nu \longmapsto \langle G_3\rangle u \mid \nu$, where $G_3 = G_1 \sqcap G_2$. We only have to prove that $G_1 \sqcap G_2 \sqsubseteq G_2$ which is immediate from Proposition 82. ☐

Proposition 112 *(Monotonicity of the evolving heap). If $t^G \mid \nu \longmapsto t'^G \mid \nu'$, then $\forall o^{G'}_{\mathsf{m}} \in dom(\mu), ev(\nu'(o^{G'}_{\mathsf{m}})) \sqsubseteq ev(\nu(o^{G'}_{\mathsf{m}}))$.*

Proof. We proceed by induction on $t^G \mid \nu \longmapsto t'^G \mid \nu'$. We only illustrate representative cases.

Case *(RE and r4).* Then $\mathsf{ref}^{G_2}_z\ \langle G_1\rangle u \mid \nu \longmapsto \langle \mathsf{Ref}\ G_1\rangle o^{G_2}_z :: \mathsf{Ref}\ G_2 \mid \nu'$, where $\nu' = \nu[o^{G_2}_z \mapsto \langle G_1\rangle u :: G_2]$ and $o^{G_2}_z \notin dom(\nu)$. But then $\forall o^{G'}_z \in dom(\nu), \nu(o^{G'}_z) = \nu'(o^{G'}_z)$ and the result holds immediately.

Case *(RE and r6, $t^G = \langle \mathsf{Ref}\ G_1\rangle o^{G_4}_{\mathsf{m}} :=^{G_3} \langle G_2\rangle u$).* Then $t^G \mid \nu \longmapsto \mathsf{unit} \mid \nu[o^{G_4}_{\mathsf{m}} \mapsto \langle G'\rangle(\langle G_2 \sqcap G_1\rangle u :: G_4) :: G_4]$, where $\nu(o^{G_4}_{\mathsf{m}}) = \langle G'\rangle u :: G_4$. Then we have to prove that $\langle G'\rangle \sqsubseteq \langle G'\rangle$, but is trivial.

Case *(RE and r6, $t^G = \langle \mathsf{Ref}\ G_1\rangle o^{G_4}_z :=^{G_3} \langle G_2\rangle u, z \neq \mathsf{m}$).* The result is immediate as the updated location is not monotone.

Case *($R\nu$).* We know that $t \mid \nu \longmapsto t \mid \nu'[o^{G'}_z \mapsto et' :: G']$, where $\nu(o^{G'}_z) = et :: G'$ and $et \mid \nu \longrightarrow_c et' \mid \nu'$, and $ev(\nu(o^{G}_z)) = ev(\nu'(o^{G}_z))$. By Lemma 111, $et' \sqsubseteq et$ and $\forall o^{G'}_{\mathsf{m}} \in dom(\nu), ev(\nu'(o^{G'}_{\mathsf{m}})) \sqsubseteq ev(\nu(o^{G'}_{\mathsf{m}}))$. We have to prove that $\forall o^{G'}_{\mathsf{m}} \in dom(\nu), ev(\nu'[o^{G'}_z \mapsto et' :: G'](o^{G'}_{\mathsf{m}})) \sqsubseteq ev(\nu(o^{G'}_{\mathsf{m}}))$, which means that we only have to show that $et' :: G' \sqsubseteq et :: G'$, but as $et' \sqsubseteq et$, the result is immediate. ☐

Proposition 113 *(Monotonicity of the heap). If $t^G \mid \mu \longmapsto^* t'^G \mid \mu'$, then $\forall o^{G'}_{\mathsf{m}} \in dom(\mu), \mu(o^{G'}_{\mathsf{m}}) = \varepsilon u :: G'$, then $\mu'(o^{G'}_{\mathsf{m}}) = \varepsilon' u' :: G'$ and $\varepsilon' \sqsubseteq \varepsilon$.*

Proof. By induction on $t^G \mid \mu \longmapsto^* t'^G \mid \mu'$ and Proposition 24. $\square$

Proposition 114 *($\longrightarrow$ is well defined). If $t^G \vdash \mu$ and $t^G \mid \mu \longrightarrow r$, then $r \in \textsc{Config}_G \cup \{\textbf{error}\}$, and if $r = t'^G \mid \mu$, then also $t'^G \vdash \nu$ and $dom(\mu) \subseteq dom(\nu')$.*

Proof. By induction on the structure of a derivation of $t^G \longrightarrow r$, considering the last rule used in the derivation. The proof is analogous to some cases considered in Proposition 24. We only illustrate representative cases.

Case *(r4).* Then $t^G = \mathsf{ref}_z^{G_2}\ \langle G_1\rangle u$. Then

$$(\text{IGref})\frac{u \in \mathbb{T}[G_1] \qquad \langle G_1\rangle \vdash G_1' \sim G_2}{\mathsf{ref}_z^{G_2}\ \langle G_1\rangle u \in \mathbb{T}[\mathsf{Ref}\ G_2]}$$

Then

$$\mathsf{ref}_z^{G_2}\ \langle G_1\rangle u \mid \mu \longrightarrow \langle \mathsf{Ref}\ G_2\rangle o_z^{G_2} :: \mathsf{Ref}\ G_2 \mid \mu[o^{G_2} \mapsto \langle G_1\rangle u :: G_2]$$

where $o_z^{G_2} \notin dom(\mu)$. But as $\langle G_1\rangle u :: G_2 \in \mathbb{T}[G_2]$, then $\langle \mathsf{Ref}\ G_2\rangle o_z^{G_2} :: \mathsf{Ref}\ G_2 \vdash \mu[o_z^{G_2} \mapsto \langle G_1\rangle u :: G_2]$. Also as $\langle \mathsf{Ref}\ G_2\rangle \vdash \mathsf{Ref}\ G_2 \sim \mathsf{Ref}\ G_2$, $\langle \mathsf{Ref}\ G_2\rangle o_z^{G_2} :: \mathsf{Ref}\ G_2 \in \mathbb{T}[\mathsf{Ref}\ G_2]$ and the result holds.

Case *(r6).* Then $t^G = \varepsilon_1 o_z^{G_1} :=^{G_3} \varepsilon_2 u$. Then

$$(\text{IGasgn})\frac{\begin{array}{cc} o^{G_1} \in \mathbb{T}[\mathsf{Ref}\ G_1] & \varepsilon_1 \vdash \mathsf{Ref}\ G_1 \sim \mathsf{Ref}\ G_3 \\ u \in \mathbb{T}[G_2] & \varepsilon_2 \vdash G_2 \sim G_3 \end{array}}{\varepsilon_1 o_z^{G_1} :=^{G_3} \varepsilon_2 u \in \mathbb{T}[\mathsf{Unit}]}$$

Suppose $\mu(o_z^{G_1}) = \varepsilon_3 u' :: G_1$, and $z = \mathsf{m}$. If $\varepsilon' = (\varepsilon_2 \sqcap \varepsilon_3 \sqcap iref(\varepsilon_1))$ is not defined, then $t^G \longrightarrow \textbf{error}$, and then the result hold immediately. We know that $v = \varepsilon_2 u :: G_3 \in \mathbb{T}[G_3]$. Also $iref(\varepsilon_1) \vdash G_3 \sim G_1$, and $\varepsilon_3 \vdash G_{u'} \sim G_1$, and $\varepsilon_3 \vdash G_1 \sim G_{u'}$, therefore $\varepsilon_3 \vdash G_1 \sim G_1$, then $iref(\varepsilon_1) \circ^{=} \varepsilon_3 \vdash G_3 \sim G_1$, then $t = (iref(\varepsilon_1) \circ^{=} \varepsilon_3) v :: G_1 \in \mathbb{T}[G_1]$. If $z \neq \mathsf{m}$, then by using arguments analogous to the other case we know that $t = iref(\varepsilon_1) v :: G_1 \in \mathbb{T}[G_1]$. Therefore as $freeLocs(\mathsf{unit}) = \emptyset \subseteq dom(\mu)$, we know from $t^G \vdash \mu$ that $\forall o^{G'} \in dom(\mu), \mu(o^{G'}) \in \mathbb{T}[G']$, and as $t \in \mathbb{T}[G_1]$, therefore $\mathsf{unit} \vdash \mu[o^{G_1} \mapsto t]$. Also

$$\frac{\theta(\mathsf{unit}) = \mathsf{Unit}}{\mathsf{unit} \in \mathbb{T}[\mathsf{Unit}]}$$

and the result holds. $\square$

Proposition 115 *($\longrightarrow_c$ is well defined). If $\varepsilon t \in \textsc{EvTerm}_G$, $t \vdash \nu$ and $\varepsilon t \mid \nu \longrightarrow_c r$, then $r \in (\textsc{EvTerm}_G \times \textsc{EvolvingStore}) \cup \{\textbf{error}\}$, and if $r = \varepsilon' t' \mid \nu'$, then also $t' \vdash \nu'$ and $dom(\nu) \subseteq dom(\nu')$.*

Proof. As $\varepsilon t \mid \nu \longrightarrow_c r$, then $t = \varepsilon_u u :: G'$, and as $\varepsilon(\varepsilon_u u :: G') \in \textsc{EvTerm}_G$ then $\varepsilon_u \vdash G_u \sim G'$ and $\varepsilon \vdash G' \sim G$, for some G_u, G', with $u \in \mathbb{T}[G_u]$. If $\varepsilon_u \sqcap \varepsilon$ is not defined then $r = \textbf{error}$ and the result holds immediately. Let us assume that $\varepsilon_c = \varepsilon_u \sqcap \varepsilon$ is defined. Then $\varepsilon t \mid \nu \longrightarrow_c \varepsilon_c u \mid \nu'$. By definition of consistent transitivity $\varepsilon_c \vdash G_u \sim G$, therefore $t' = \varepsilon_c u \in \textsc{EvTerm}_G$. If $u \neq o_{\mathsf{m}}^{G_u'}$, or if $(u = o_{\mathsf{m}}^{G_u'}$ and $\nu(u) = iref(\varepsilon_c)) t'' :: G_u')$, then $\nu = \nu'$ and the result holds. Let us assume that $u = o_{\mathsf{m}}^{G_u'}$, $\nu(u) = \varepsilon'' t'' :: G_u'$, and $\varepsilon'' \neq iref(\varepsilon_c)$, then $\nu' = \nu[u \mapsto \varepsilon_c' \nu(u) :: G_u']$, where $\varepsilon_c' = \varepsilon'' \sqcap iref(\varepsilon_c)$. Also $G_u = \mathsf{Ref}\ G_u'$ and $G = \mathsf{Ref}\ G''$, for some G''. No new locations are created then $dom(\nu) = dom(\nu') \Rightarrow dom(\nu) \subseteq dom(\nu')$. We have the obligation to prove that $u \vdash \nu'$. As the domains are the same it is easy to see that $freeLocs(u) \subseteq dom(\nu')$. Then we only have to prove that $\varepsilon_c' \nu(u) :: G_u \in \mathbb{T}[G_u]$. But as $\varepsilon_c \vdash \mathsf{Ref}\ G_u' \sim \mathsf{Ref}\ G''$, then $iref(\varepsilon_c) \vdash G_u' \sim G''$, and also $iref(\varepsilon_c) \vdash G'' \sim G_u'$, therefore $iref(\varepsilon_c) \vdash G_u' \sim G_u'$. Similarly $\varepsilon'' \vdash G_u' \sim G_u'$, thus $\varepsilon_c' \vdash G_u' \sim G_u'$. Finally then $\varepsilon_c' \nu(u) :: G_u \in \mathbb{T}[G_u]$ and the result holds. $\square$

Proposition 116 *($\longmapsto$ is well defined). If $t^G \vdash \nu$ and $t^G \mid \nu \longmapsto r$, then $r \in \textsc{Config}_G \cup \{\textbf{error}\}$, and if $r = t'^G \mid \nu'$, then also $t'^G \vdash \nu'$ and $dom(\nu) \subseteq dom(\nu')$.*

Proof. By induction on the structure of a derivation of $t^G \longmapsto r$.
We proceed almost identical to 42, therefore we only illustrate main differences.

Case *(RF).* Let $\textsc{EvTerm}_{G_2}$ be notation for the family of evidence terms εt^{G_1} such that $\varepsilon \vdash G_1 \sim G_2$. Then $t^G = F[et]$, $F[et] \in \mathbb{T}[G]$, and $F : \textsc{EvTerm}_{G_x} \rightarrow \mathbb{T}[G]$, and $et \mid \nu \longrightarrow_c et' \mid \nu'$. By Lemma 115, $et' \in \textsc{EvTerm}_{G_x}$, $et' \vdash \nu'$, and $dom(\nu) \subseteq dom(\nu')$. Then $F[et'] \in \mathbb{T}[G]$, and as $freeLocs(et) = freeLocs(et')$ we can conclude that $F[et'] \vdash \nu$.

Case *(Rν)*. We know that $t \mid \nu \longmapsto t \mid \nu'[o_z^{G'} \mapsto et' :: G']$, where $\nu(o_z^{G'}) = et :: G'$ and $et \mid \nu \longrightarrow_c et' \mid \nu'$, and $ev(\nu(o_z^G)) = ev(\nu'(o_z^G))$. By Lemma 115, $et' \in \text{EvTerm}_{G'}$, $et' \vdash \nu'$, and $dom(\nu) \subseteq dom(\nu')$. As $dom(\nu) \subseteq dom(\nu')$ then $freeLocs(t) \subseteq dom(\nu')$. As $et' \vdash \nu'$, we know that $\forall o^{G''} \in dom(\nu'), \nu'(o^{G''}) \in \mathbb{T}[G]$. Then we only have to prove that $et' :: G'$, but we know that $et' \in \text{EvTerm}_{G'}$, therefore $et' :: G' \in \mathbb{T}[G']$ and the result holds.

Case *(νErr)*. Trivial as $t \mid \nu \longmapsto$ **error**. □

To define the dynamic gradual guarantee, first we have to extend the notion of precision to evolving stores. Note that rule $(r7)$ propagates casts into the store based on a type test, which may seem to jeopardize the dynamic gradual guarantee. Consider two terms and store in the precision relation, one of the two terms can reduce to a new term and evolving store (that needs to be reduced) whereas the other to another term and store (that does not need to be reduced). Therefore to maintain the precision relation in lock step, we define precision between evolving stores as follows:

$$flat(\varepsilon_2 t^G :: G) = \varepsilon_1 \circ^{=} flat(t^G) \qquad flat(\varepsilon_1 u :: G) = \varepsilon_1$$

$$uval(\varepsilon_2 t^G :: G) = uval(t) \qquad uval(\varepsilon_1 u :: G) = u$$

$$(\sqsubseteq_\nu)\ \frac{\begin{array}{c}\forall o^{G_1} \in dom(\nu_1).\exists o^{G_2} \in dom(\nu_2)\ s.t. \\ \Omega \vdash o^{G_1} \sqsubseteq o^{G_2} \quad G_1 \sqsubseteq G_2 \quad uval(\nu_1(o^{G_1})) \sqsubseteq uval(\nu_2(o^{G_2})) \\ flat(\nu_1(o^{G_1}))\text{ is defined} \Rightarrow flat(\nu_1(o^{G_1})) \sqsubseteq flat(\nu_2(o^{G_2}))\end{array}}{\Omega \vdash \nu_1 \sqsubseteq \nu_2}$$

Note that (1) if $G \sqsubseteq G'$, $uval(t^G) \sqsubseteq uval(t^{G'})$, but consistent transitivity is not defined in *flat* of tG, then the relation hold, and (2) if both evolving stores are stores, then this definition coincides with the precision relation of stores defined for $\lambda^{\varepsilon}_{\widetilde{\text{REF}}}$.

Proposition 117 *(Dynamic guarantee for $\longrightarrow$). Suppose $\Omega \vdash t_1^{G_1} \sqsubseteq t_1^{G_2}$ and $\mu_1 \sqsubseteq \mu_2$. If $t_1^{G_1} \mid \mu_1 \longrightarrow t_2^{G_1} \mid \nu_1$ then $t_1^{G_2} \mid \mu_2 \longrightarrow t_2^{G_2} \mid \nu_2$, where $\Omega' \vdash t_2^{G_1} \sqsubseteq t_2^{G_2}$, $\nu_1 \sqsubseteq \nu_2$ for some $\Omega' \supseteq \Omega$.*

Proof. By induction on reduction $t_1^{G_1} \mid \mu_1 \longrightarrow t_2^{G_1} \mid \nu_1$. We proceed almost identical to 66, therefore we only illustrate main differences. For simplicity we omit the $\Omega \vdash$ notation on precision relations when it is not relevant for the argument.

Case *(r2)*. We know that $t_1^{G_1} = \varepsilon_{11}(\lambda x^{G_{11}}.t^{G_{12}})\ @^{G_1 \longrightarrow G_2}\ \varepsilon_{12} u$ then by $(\sqsubseteq_{app})$ $t_1^{G_2}$ must have the form $t_1^{G_2} = \varepsilon_{21}(\lambda x^{G_{21}}.t^{G_{22}})\ @^{G_3 \longrightarrow G_4}\ \varepsilon_{22} u_2$ for some $\varepsilon_{21}, x^{G_{21}}, t^{G_{22}}, G_3, G_4, \varepsilon_{22}$ and u_2.

Let us pose $\varepsilon_1 = \varepsilon_{12} \circ^{=} idom(\varepsilon_{11})$. Then $t_1^{G_1} \mid \mu_1 \longrightarrow icod(\varepsilon_{11})t_1' :: G_2 \mid \mu_1$ with $t_1' = [t_1'/x^{G_{11}}]t^{G_{12}}$, and $t_1' = idom(\varepsilon_{11})(\varepsilon_{12}u_1 :: G_1) :: G_{11}$.

Also, by 64, $\varepsilon_2 = \varepsilon_{22} \circ^{=} idom(\varepsilon_{21})$ is defined. Then $t_1^{G_2} \mid \mu_2 \longrightarrow icod(\varepsilon_{21})t_2' :: G_4 \mid \mu_2$ with $t_2' = [t_2'/x^{G_{21}}]t^{G_{22}}$, and $t_2' = idom(\varepsilon_{21})(\varepsilon_{22}u_2 :: G_3) :: G_{21}$.

As $\Omega \vdash t_1^{G_1} \sqsubseteq t_1^{G_2}$, then $u_1 \sqsubseteq u_2$, $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$ and $idom(\varepsilon_{11}) \sqsubseteq idom(\varepsilon_{21})$ as well. Then $t_1' \sqsubseteq t_2'$ by $(\sqsubseteq_{::})$.

We also know by $(\sqsubseteq_{APP})$ and $(\sqsubseteq_\lambda)$ that $\Omega \cup \{x^{G_{21}} \sqsubseteq x^{G_{21}}\} \vdash t^{G_{12}} \sqsubseteq t^{G_{22}}$. By Substitution preserves precision (Proposition 63) $t_1' \sqsubseteq t_2'$, therefore $icod(\varepsilon_{11})t_1' :: G_2 \sqsubseteq icod(\varepsilon_{21})t_2' :: G_4$ by $(\sqsubseteq_{::})$. Then $t_2^{G_1} \sqsubseteq t_2^{G_2}$.

Case *(r4)*. We know that $t_1^{G_1} = \mathsf{ref}^{G_1'}\ \varepsilon_1 u_1$ where $G_1 = \mathsf{Ref}\ G_1'$, then by $(\sqsubseteq_{REF})$ $t_1^{G_2}$ must have the form $t_1^{G_2} = \mathsf{ref}^{G_2'}\ \varepsilon_2 u_2$ for some ε_2, u_2, G_2' such that $\varepsilon_1 \sqsubseteq \varepsilon_2, u_1 \sqsubseteq u_2$, and $G_1' \sqsubseteq G_2'$.

Then $t_1^{G_1} \mid \mu_1 \longrightarrow \langle \mathsf{Ref}\ G_1' \rangle o_z^{G_1'} :: \mathsf{Ref}\ G_1' \mid \mu_1[o_z^{G_1'} \mapsto \varepsilon_1 u_1 :: G_1']$.

Also, $t_1^{G_2} \mid \mu_2 \longrightarrow \langle \mathsf{Ref}\ G_2' \rangle o_z^{G_2'} :: \mathsf{Ref}\ G_2' \mid \mu_2[o_z^{G_2'} \mapsto \varepsilon_2 u_2 :: G_2']$.

Then by $(\sqsubseteq_{::})$, $\varepsilon_1 u_1 :: G_1' \sqsubseteq \varepsilon_2 u_2 :: G_2'$, and then $\mu_1[o_z^{G_1'} \mapsto \varepsilon_1 u_1 :: G_1'] \sqsubseteq \mu_2[o_z^{G_2'} \mapsto \varepsilon_2 u_2 :: G_2']$. Also by $(\sqsubseteq_o)$, as $G_1' \sqsubseteq G_2'$ and by $(\sqsubseteq_{::})$, $\langle \mathsf{Ref}\ G_1' \rangle o_z^{G_1'} :: \mathsf{Ref}\ G_1' \sqsubseteq \langle \mathsf{Ref}\ G_2' \rangle o_z^{G_2'} :: \mathsf{Ref}\ G_2'$ and the result holds.

Case *(r6 where $z = \mathsf{m}$)*. We know that $t_1^{G_1} = \varepsilon_{11} o_z^{G_{11}} :=^{G_{12}} \varepsilon_{12} u_1$ where $G_1 = \mathsf{Unit}$, then by $(\sqsubseteq_{:=})$ $t_1^{G_2}$ must have the form $t_1^{G_2} = \varepsilon_{21} o_z^{G_{21}} :=^{G_{22}} \varepsilon_{22} u_2$ for some $\varepsilon_{21}, \varepsilon_{22}, u_2, G_{21}, G_{22}$ such that $\varepsilon_{11} \sqsubseteq \varepsilon_{21}, \varepsilon_{12} \sqsubseteq \varepsilon_{22}, u_1 \sqsubseteq u_2, G_{11} \sqsubseteq G_{21}, G_{12} \sqsubseteq G_{22}$.

Suppose $\mu_1(o_z^{G_{11}}) = \varepsilon_1' u_1' :: G_{11}$, and as $\mu_1 \sqsubseteq \mu_2$, then $\mu_2(o_z^{G_{21}}) = \varepsilon_2' u_2' :: G_{21}$, such that $\varepsilon_1' \sqsubseteq \varepsilon_2'$, $u_1' \sqsubseteq u_2'$. Let us pose $\varepsilon_1 = iref(\varepsilon_{11}) \circ^{=} \varepsilon_1'$. Then $t_1^{G_1} \mid \mu_1 \longrightarrow \mathsf{unit} \mid \mu_1[o_z^{G_{11}} \mapsto \langle \varepsilon_1 \rangle(\varepsilon_{12}u_1 :: G_{11}) :: G_{11}]$.

By inspection of evidence and inversion lemma, as $\varepsilon_{11} \sqsubseteq \varepsilon_{21}$ then $iref(\varepsilon_{11}) \sqsubseteq iref(\varepsilon_{21})$. Also, by Lemma 64, $\varepsilon_2 = iref(\varepsilon_{21}) \circ^{=} \varepsilon_2'$ is defined (similarly $\varepsilon_{22} \circ^{=} \varepsilon_2$ is also defined) and $\varepsilon_1 \sqsubseteq \varepsilon_2$. Then, $t_1^{G_2} \mid \mu_2 \longrightarrow \mathsf{unit} \mid \mu_2[o_z^{G_{21}} \mapsto \langle \varepsilon_2 \rangle(\varepsilon_{22}u_2 :: G_{21}) :: G_{21}]$.

Now we know that $\varepsilon_{12} \circ^{=} \varepsilon_1$ is defined, and as $\varepsilon_1 \sqsubseteq \varepsilon_2$ and $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$, by Lemma 64, $\varepsilon_{22} \circ^{=} \varepsilon_2$ is also defined. Therefore $flat(\langle\varepsilon_1\rangle(\varepsilon_{12} u_2 :: G_{11}) :: G_{11}]) \sqsubseteq flat(\langle\varepsilon_2\rangle(\varepsilon_{22} u_2 :: G_{21}) :: G_{21}])$ and the result holds □

Definition 16.

$$\vdash_{\mathsf{m}} (t, \nu) \iff \text{if } \varepsilon o^{G}_{\mathsf{m}} \text{ appears in } (t, \nu), \text{ either } flat(\nu(o^{G}_{\mathsf{m}})) \text{ is not defined, or } flat(\nu(o^{G}_{\mathsf{m}})) \sqsubseteq iref(\varepsilon).$$

Lemma 118. *If $\varepsilon_1 \sqsubseteq \varepsilon_2$ and $\varepsilon_1 \circ^{=} \varepsilon_2$ is defined, then $\varepsilon_1 \circ^{=} \varepsilon_2 \sqsubseteq \varepsilon_2$*

Proof. We know that $\varepsilon_2 \circ^{=} \varepsilon_2 = \varepsilon_2$. Then by Proposition 64, $\varepsilon_1 \circ^{=} \varepsilon_2 \sqsubseteq \varepsilon_2 \circ^{=} \varepsilon_2$, and the result holds. □

Lemma 119. *If $\varepsilon_1 \sqsubseteq \varepsilon_2$ and $\varepsilon_1 \circ^{=} \varepsilon_3$ is defined, then $\varepsilon_1 \circ^{=} \varepsilon_3 \sqsubseteq \varepsilon_2$*

Proof. Direct by formal definition of meet and precision using the concretization function. □

Lemma 120. *If $\varepsilon_1 \sqsubseteq \varepsilon_2$, $\varepsilon_1 \sqsubseteq \varepsilon_3$, then $\varepsilon_2 \circ^{=} \varepsilon_3$ is defined, and $\varepsilon_1 \sqsubseteq \varepsilon_2 \circ^{=} \varepsilon_3$*

Proof. By induction on ε_1. □

Lemma 121 *(Monotonic well-formedness preservation). If $\vdash_{\mathsf{m}} (t, \nu)$ and $t \mid \nu \longmapsto t' \mid \nu'$, then $\vdash_{\mathsf{m}} (t', \nu')$.*

Proof. By induction on $t \mid \nu \longmapsto t' \mid \nu'$. We only consider interesting cases where references are involved.

Case *(r2).* Then

$$F[(\langle G'_{11} \to G'_{12}\rangle(\lambda x^{G_{11}}.t))@^{G_1 \to G_2} (\langle G'_2\rangle u)] \| \mu \longmapsto F\langle G'_{12}\rangle([v/x^{G_{11}}]t) :: G_2] \| \nu$$

where $\langle G'_{11}\rangle(\langle G'_2\rangle u :: G_1) :: G_{11} \| \mu \longmapsto v \mid \nu$.

We know that $\vdash_{\mathsf{m}} (\langle G'_2\rangle u, \mu)$, then the result follows from induction hypothesis on $\langle G'_{11}\rangle(\langle G'_2\rangle u :: G_1) :: G_{11} \| \mu \longmapsto v \mid \nu$.

Case *(r4).* Then

$$F[\mathsf{ref}^{G_2}_{\mathsf{m}}\ \langle G_1\rangle u] \mid \mu \longmapsto F[\langle \mathsf{Ref}\ G_2\rangle o^{G_2}_{\mathsf{m}} :: \mathsf{Ref}\ G_2] \mid \mu[o^{G_2}_{\mathsf{m}} \mapsto \langle G_1\rangle u :: G_2]$$

where $\langle G'_{11}\rangle(\langle G'_2\rangle u :: G_1) :: G_{11} \| \mu \longmapsto v \mid \nu$.

Suppose $u \in \mathbb{T}[G']$. We know that $\langle G_1\rangle \vdash G' \sim G_2$, therefore $G_1 \sqsubseteq G_2$, and thus $\langle G_1\rangle \sqsubseteq \langle G_2\rangle$. We have to prove that $\langle G_1\rangle \sqsubseteq iref(\langle \mathsf{Ref}\ G_2\rangle) = \langle G_2\rangle$ which follows immediately. The result follows because from $\vdash_{\mathsf{m}} (t, \nu)$ we know that if $u = o^{G'}_{\mathsf{m}}$, then $flat(\mu(u)) \sqsubseteq iref\langle G_1\rangle$.

Case *(r5).* Then

$$F[!^{G}(\langle \mathsf{Ref}\ G_1\rangle o^{G_2}_{\mathsf{m}})] \mid \mu \longmapsto F[\langle G_1\rangle v :: G] \mid \mu$$

where $v = \mu(o^{G_2}_{\mathsf{m}})$. The result is immediately as we know that $\vdash_{\mathsf{m}} (v, \mu)$.

Case *(r6).* Then

$$F[\langle \mathsf{Ref}\ G_1\rangle o^{G}_{\mathsf{m}} :=^{G_3} \langle G_2\rangle u] \mid \mu \longmapsto F[\mathsf{unit}] \mid \mu[o^{G}_{\mathsf{m}} \mapsto t']$$

where $\mu(o^{G}_{\mathsf{m}}) = \langle G'\rangle u' :: G, t' = \langle G_1 \sqcap G'\rangle(\langle G_2\rangle u :: G_3) :: G$ We have two obligations: (1) if $\varepsilon' o^{G}_{\mathsf{m}}$ appears in $cod(\mu)$, then $flat(t') \sqsubseteq iref(\varepsilon')$, and (2) if u is a monotonic location then $iref(\langle G_2\rangle) \sqsubseteq flat(\mu(u))$.

Let us prove (1). We know by $\vdash_{\mathsf{m}} (t, \mu)$, that if $\varepsilon' o^{G}_{\mathsf{m}}$ appears in $cod(\mu)$, then $\langle G'\rangle \sqsubseteq iref(\varepsilon')$. If $flat(t') = \langle G_2\rangle \circ^{=} \langle G_1 \sqcap G'\rangle$ is defined (if not defined the result is trivial) then by Proposition 118, $flat(t') \sqsubseteq \langle G'\rangle$, and therefore $flat(t') \sqsubseteq iref(\varepsilon')$.

Now to prove (2), we already know that $\vdash_{\mathsf{m}} (\langle G_2\rangle u :: G_3, \mu)$, then it is trivial to see that $\vdash_{\mathsf{m}} (t', \mu)$ and the result holds.

Case *(r7).* Then

$$\langle G_2\rangle(\langle G_1\rangle o^{G_5}_{\mathsf{m}} :: G) \| \nu \longrightarrow_c \langle G_3\rangle o^{G_5}_{\mathsf{m}} \mid \nu[o^{G_5}_{\mathsf{m}} \mapsto \langle G_4\rangle \nu(o^{G_5}_{\mathsf{m}}) :: G_5] \qquad G' \neq \widetilde{tref}(G_3)$$

where $ev(\nu(o^{G_5}_{\mathsf{m}})) = \langle G'\rangle, G_3 = G_1 \sqcap G_2$, and $G_4 = \widetilde{tref}(G_3) \sqcap G'$, by either $(R\nu)$ or (RF). Let us prove first that $\vdash_{\mathsf{m}} (\langle G_3\rangle o^{G_5}_{\mathsf{m}}, \nu[o^{G_5}_{\mathsf{m}} \mapsto \langle G_4\rangle \nu(o^{G_5}_{\mathsf{m}}) :: G_5])$.

Then we have to prove that

1. If $\varepsilon' o_m^{G_5}$ appears in $cod(\nu)$, then $flat(\langle G_4\rangle \nu(o_m^{G_5}) :: G_5) \sqsubseteq iref(\varepsilon')$. Notice that $flat(\langle G_4\rangle \nu(o_m^{G_5}) :: G_5) = \langle G'' \sqcap G_4\rangle$, for some $G'' \sqsubseteq G'$ (if not defined the result follows). We know from $\vdash_m (t, \nu)$ that $flat(\nu(o_m^{G_5})) = \langle G''\rangle \sqsubseteq iref(\varepsilon')$, then it by Proposition 118, $\langle G'' \sqcap G_4\rangle \sqsubseteq \langle G''\rangle \sqsubseteq iref(\varepsilon')$ and the result follows.
2. $\vdash_m (\langle G_4\rangle \nu(o_m^{G_5}) :: G_5, \nu')$. We know from $\vdash_m (t, \nu)$ that if $\varepsilon' o_m^{G_6}$ appears in $\nu(o_m^{G_5})$, then $flat(\nu(o_m^{G_6})) \sqsubseteq iref(\varepsilon')$. Then as $\varepsilon' o_m^{G_6}$ also appears in $\langle G_4\rangle \nu(o_m^{G_5}) :: G_5$ the result follows.

Now if (Rν) is used, i.e.

$$(\mathrm{R}\nu)\frac{\nu(o_m^G) = et :: G \qquad et \mid \nu \longrightarrow_c et' \mid \nu'}{t \mid \nu \longmapsto t \mid \nu'[o_m^G \mapsto et' :: G]}$$

Then we have to prove that

1. If $\varepsilon' o_m^{G_5}$ appears in (t, ν), then $flat(\langle G_4\rangle \nu(o_m^{G_5}) :: G_5) \sqsubseteq iref(\varepsilon')$. Notice that $flat(\langle G_4\rangle \nu(o_m^{G_5}) :: G_5) = \langle G'' \sqcap G_4\rangle$, for some $G'' \sqsubseteq G'$ (if not defined the result follows). We know from $\vdash_m (t, \nu)$ that $flat(\nu(o_m^{G_5})) = \langle G''\rangle \sqsubseteq iref(\varepsilon')$, then by Proposition 118, $\langle G'' \sqcap G_4\rangle \sqsubseteq \langle G''\rangle \sqsubseteq iref(\varepsilon')$ and the result follows.
2. If $\varepsilon' o_m^G$ appears in (t, ν), then $flat(\langle G_3\rangle o_m^{G_5} :: G) = \langle G_3\rangle \sqsubseteq iref(\varepsilon')$. We know from $\vdash_m (t, \nu)$ that $flat(\nu(o_m^G)) = \langle G_1\rangle \sqsubseteq iref(\varepsilon')$, then by Proposition 118, $\langle G_1 \sqcap G_2\rangle \sqsubseteq \langle G_1\rangle \sqsubseteq iref(\varepsilon')$ and the result follows.
3. $flat(\langle G_4\rangle \nu(o_m^{G_5}) :: G_5) \sqsubseteq iref(\langle G_3\rangle)$. We know that $flat(\nu(o_m^{G_5})) \sqsubseteq iref(\langle G_1\rangle)$. Then by Proposition 118 $flat(\langle \widetilde{tref}(G_2)\rangle \nu(o_m^{G_5}) :: G_5) = flat(iref(\langle G_2\rangle) \nu(o_m^{G_5}) :: G_5) \sqsubseteq \langle \widetilde{tref}(G_1 \sqcap G_2)\rangle = iref(\langle G_3\rangle)$. Also,

$$flat(\langle G_4\rangle \nu(o_m^{G_5}) :: G_5) = flat(\langle \widetilde{tref}(G_1 \sqcap G_2) \sqcap G'\rangle \nu(o_m^{G_5}) :: G_5) \sqsubseteq flat(iref(\langle G_2\rangle) \nu(o_m^{G_5}) :: G_5)$$

therefore from Proposition 118 $flat(\langle G_4\rangle \nu(o_m^{G_5}) :: G_5) \sqsubseteq flat(\nu(o_m^{G_5})) \sqsubseteq iref(\langle G_3\rangle)$ and the result holds.

Case (RF) is analogous to (Rν) (and simpler) . □

Lemma 122. *If $iref(iref(G)) \sqcap G$ is defined, then $G \sqsubseteq \widetilde{tref}(\widetilde{tref}(G))$.*

Proof. We proceed by induction on $\widetilde{tref}(\widetilde{tref}(G))$.

Case *($G = ?$).* Then we have to prove that $? \sqsubseteq ?$, which is direct.

Case *($G = \mathsf{Ref}\ ?$).* Then we have to prove that $\mathsf{Ref}\ ? \sqsubseteq ?$, which is direct.

Case *($G = \mathsf{Ref}\ \mathsf{Ref}\ G' = \mathsf{Ref}\ {}^2G'$).* We have to prove that $\mathsf{Ref}\ {}^2G' \sqsubseteq G'$. We analyze two possible cases. If $G' = ?$, then the result is trivial as $\mathsf{Ref}\ {}^2? \sqsubseteq ?$. If $G' = \mathsf{Ref}\ G''$ for some G'', then we have to prove that $\mathsf{Ref}\ {}^3G'' \sqsubseteq \mathsf{Ref}\ G''$, but by inspection of $\sqsubseteq$, it is equivalent to prove that $\mathsf{Ref}\ {}^2G'' \sqsubseteq G''$, which is equivalent to prove that $\mathsf{Ref}\ {}^2G'' \sqsubseteq iref^2(\mathsf{Ref}\ {}^2G'')$. We know that $\mathsf{Ref}\ G'' \sqcap \mathsf{Ref}\ {}^3G''$ is defined, therefore by definition of $\sqcap$, it must be the case that $G'' \sqcap \mathsf{Ref}\ {}^2G''$ is defined. Then by induction hypothesis $\mathsf{Ref}\ {}^2G'' \sqsubseteq iref^2(\mathsf{Ref}\ {}^2G'')$, and the result holds. □

Lemma 123. *If $\vdash_m (t, \mu[o_m^G \mapsto \varepsilon_1(\varepsilon_2 u :: G) :: G])$, $t \mid \mu[o_m^G \mapsto \varepsilon_1(\varepsilon_2 u :: G) :: G] \longmapsto t \mid \nu[o_m^G \mapsto \varepsilon_3 u :: G]$, then $t \mid \nu[o_m^G \mapsto \varepsilon_3 u :: G] \not\longmapsto^* t \mid \nu'[o_m^G \mapsto \varepsilon_4(\varepsilon_5 u :: G) :: G]$.*

Proof. We proceed by contradiction. Without losing generality, let us suppose that there is the following cycle: $\nu(o_m^{G_1}) = \varepsilon(\varepsilon_1 o_m^{G_2} :: G_1) :: G_1, \nu(o_m^{G_2}) = \varepsilon_2 o_m^{G_1} :: G_2$. Then suppose

$$\begin{aligned} t \mid \nu &\longmapsto t \mid \mu[o_m^{G_1} \mapsto (\varepsilon_1 \circ \varepsilon) o_m^{G_2} :: G_1, o_m^{G_2} \mapsto iref(\varepsilon_1 \circ \varepsilon)(\varepsilon_2 o_m^{G_1} :: G_2) :: G_2] \\ &\longmapsto t \mid \mu[o_m^{G_1} \mapsto iref(\varepsilon')((\varepsilon_1 \circ \varepsilon) o_m^{G_2} :: G_1) :: G_1, o_m^{G_2} \mapsto \varepsilon' o_m^{G_1} :: G_2] \end{aligned}$$

where $\varepsilon' = \varepsilon_2 \circ iref(\varepsilon_1 \circ \varepsilon)$. From last step of reduction we know that $(\varepsilon_1 \circ \varepsilon) \not\sqsubseteq iref(\varepsilon_2 \circ iref(\varepsilon_1 \circ \varepsilon))$. From $\vdash_m (t, \nu)$ and Lemma 26, we know that $(\varepsilon_1 \circ \varepsilon) \sqsubseteq iref(\varepsilon_2)$. Also as $iref(\varepsilon_2 \circ iref(\varepsilon_1 \circ \varepsilon)) \circ (\varepsilon_1 \circ \varepsilon) = (iref(\varepsilon_2) \circ iref(iref(\varepsilon_1 \circ \varepsilon))) \circ (\varepsilon_1 \circ \varepsilon) = iref(\varepsilon_2) \circ (iref(iref(\varepsilon_1 \circ \varepsilon)) \circ (\varepsilon_1 \circ \varepsilon))$ is defined (using Lemma 89), then $iref(iref(\varepsilon_1 \circ \varepsilon)) \circ (\varepsilon_1 \circ \varepsilon)$ is defined. Then by Lemma 122, $(\varepsilon_1 \circ \varepsilon) \sqsubseteq iref(iref(\varepsilon_1 \circ \varepsilon))$, but we know that $(\varepsilon_1 \circ \varepsilon) \sqsubseteq iref(\varepsilon_2)$, therefore by Lemma 120, $(\varepsilon_1 \circ \varepsilon) \sqsubseteq iref(\varepsilon_2 \circ iref(\varepsilon_1 \circ \varepsilon))$ which is a contradiction and the result holds. □

Lemma 124. *If $\vdash_m (t_i, \mu_i)$, $t_1 \mid \mu_1 \sqsubseteq t_2 \mid \nu_2$, then $t_2 \mid \nu_2 \longmapsto t_2 \mid \nu_2'$, and $\mu_1 \sqsubseteq \nu_2'$.*

Proof. Let us assume $\nu_2(o_m^{G'}) = \varepsilon_2(\varepsilon_1' u' :: G') :: G'$, then

$$(\mathrm{R}\nu)\frac{\varepsilon_2(\varepsilon_1' u' :: G') \mid \nu_2 \longrightarrow_c \varepsilon_3' u' \mid \nu_2''}{t_2 \mid \nu_2 \longmapsto t_2 \mid \nu_2''[o_m^{G'} \mapsto \varepsilon_3' u' :: G']}$$

We know that if $\mu_1(o_m^G) = \varepsilon_1 u :: G$, where $u \sqsubseteq u'$ and $G \sqsubseteq G'$, then $\varepsilon_1 \sqsubseteq \mathit{flat}(\varepsilon_2(\varepsilon_1' u' :: G') :: G') \Rightarrow \varepsilon_1 \sqsubseteq \varepsilon_1' \circ^= \varepsilon_2 = \varepsilon_3$, then $\mu_1(o_m^G) \sqsubseteq \nu_2(o_m^{G'})$.

If u, u' are not locations then the result holds immediately. If $u = o_m^{G_u}$ and $u' = o_m^{G_u'}$, suppose that $\mu_1(u) = \varepsilon_4 u_4 :: G_4$, $\nu_1(u') = \varepsilon_4 u_4' :: G_4'$. We know that $\varepsilon_1 \sqsubseteq \varepsilon_3$, and thus $\mathit{iref}(\varepsilon_1) \sqsubseteq \mathit{iref}(\varepsilon_3)$. Also as $\vdash_m (t_1, \mu_1)$ we know that $\varepsilon_4 \sqsubseteq \mathit{iref}(\varepsilon_1)$, therefore $\varepsilon_4 \sqsubseteq \mathit{iref}(\varepsilon_3)$. As $\varepsilon_4 \sqsubseteq \varepsilon_4'$, then by Lemma 120 $\varepsilon_4 \sqsubseteq \mathit{iref}(\varepsilon_3) \circ^= \varepsilon_4'$. Then $\nu_2'' = \nu_2[o_m^{G_u'} \mapsto (\mathit{iref}(\varepsilon_3) \circ^= \varepsilon_4)(\varepsilon_4 u_4' :: G_4') :: G_4']$, Notice that $\mathit{flat}((\mathit{iref}(\varepsilon_3) \circ^= \varepsilon_4)(\varepsilon_4 u_4' :: G_4') :: G_4') = (\mathit{iref}(\varepsilon_3) \circ^= \varepsilon_4') \circ^= \varepsilon_4' = \mathit{iref}(\varepsilon_3) \circ^= \varepsilon_4'$ and the result holds. □

Lemma 125. *If $\vdash_m (t_i, \mu_i)$ and $t_1 \mid \mu_1 \sqsubseteq t_2 \mid \nu_2$ then $t_2 \mid \nu_2 \longmapsto^* t_2 \mid \mu_2$, such that $\mu_1 \sqsubseteq \mu_2$.*

Proof. By Lemma 124 we make sure that the precision relation holds after every step, and by Lemma 30 we notice that there are no cycles, so the biggest amount of steps before getting to a μ_2 is $\mathit{size}(\mathit{dom}(\nu_2)) - 1$. □

Lemma 126. *Let $t_1 \mid \nu_1 \sqsubseteq t_2 \mid \nu_2$. If $t_1 \mid \nu_1 \longmapsto t_1 \mid \nu_1'$, then $\nu_1' \sqsubseteq \nu_2$.*

Proof. Let us assume $\nu_1(o_m^G) = \varepsilon_2(\varepsilon_1 u :: G) :: G$, then

$$(\mathrm{R}\nu)\frac{\varepsilon_2(\varepsilon_1 u :: G) \mid \nu_2 \longrightarrow_c \varepsilon_3 u \mid \nu_1''}{t_1 \mid \nu_1 \longmapsto t_1 \mid \nu_1''[o_m^G \mapsto \varepsilon_3 u :: G]}$$

If $\nu_2(o_m^{G'}) = \varepsilon_1' u' :: G$, where $\varepsilon_1 \circ^= \varepsilon_2 \sqsubseteq \varepsilon_1'$, $u \sqsubseteq u'$ and $G \sqsubseteq G'$, then as $\mathit{flat}(\nu_2(o_m^{G'})) = \varepsilon_1'$, we have to prove that $\varepsilon_1 \circ^= \varepsilon_2 \sqsubseteq \varepsilon_1'$ which is direct. Similarly, if $\nu_2(o_m^{G'}) = \varepsilon_2' :: \varepsilon_1' u' :: G$, where $\varepsilon_1 \circ^= \varepsilon_2 \sqsubseteq \varepsilon_1' \circ^= \varepsilon_2'$, $u \sqsubseteq u'$ and $G \sqsubseteq G'$, then as $\mathit{flat}(\nu_2(o_m^{G'})) = \varepsilon_1' \circ \varepsilon_2'$, we have to prove that $\varepsilon_1 \circ^= \varepsilon_2 \sqsubseteq \varepsilon_1' \circ^= \varepsilon_2'$ which is direct.

If u, u' are not locations then the result holds immediately. If $u = o_m^{G_u}$ and $u' = o_m^{G_u'}$, suppose that $\nu_1(u) = \varepsilon_4 u_4 :: G_4$, $\nu_1(u') = \varepsilon_4 u_4' :: G_4'$. We have to prove that $(\mathit{iref}(\varepsilon_1 \circ^= \varepsilon_2) \circ^= \varepsilon_4) \circ^= \varepsilon_4 \sqsubseteq \varepsilon_4'$, but this is direct by Lemma 119. If $\nu_1(u) = \varepsilon_4 u_4 :: G_4$, and $\nu_1(u') = \varepsilon_5'(\varepsilon_4 u_4' :: G_4') :: G_4'$, then we have to prove that $(\mathit{iref}(\varepsilon_1 \circ^= \varepsilon_2) \circ^= \varepsilon_4) \circ^= \varepsilon_4 \sqsubseteq \varepsilon_4' \circ^= \varepsilon_5'$. But we know by definition of precision on evolving stores that $\varepsilon_4 \sqsubseteq \varepsilon_4' \circ^= \varepsilon_5'$ and the result holds by Lemma 119. □

Proposition 127 *(Dynamic guarantee). Suppose $\vdash_m (t_i, \nu_i)$, $t_1 \sqsubseteq t_2$ and $\nu_1 \sqsubseteq \nu_2$. If $t_1 \mid \nu_1 \longmapsto t_1' \mid \nu_1'$ then $t_2 \mid \nu_2 \longmapsto^* t_2' \mid \nu_2'$, such that $t_1' \sqsubseteq t_2'$ and $\nu_1' \sqsubseteq \nu_2'$.*

Proof. We prove the following property instead: Suppose $\Omega \vdash t_1^{G_1} \sqsubseteq t_1^{G_2}$ and $\nu_1 \sqsubseteq \nu_2$. If $t_1^{G_1} \mid \nu_1 \longmapsto t_2^{G_1} \mid \nu_1'$ then $t_1^{G_2} \mid \nu_2 \longmapsto^* t_2^{G_2} \mid \nu_2'$ where $\Omega' \vdash t_2^{G_1} \sqsubseteq t_2^{G_2}$, and $\nu_1' \sqsubseteq \nu_2'$ for some $\Omega' \supseteq \Omega$.

By induction on reduction $t_1^{G_1} \mid \nu_1 \longmapsto t_2^{G_1} \mid \nu_1'$. We proceed almost identical to 14, therefore we only illustrate main differences. For simplicity we omit the $\Omega \vdash$ notation on precision relations when it is not relevant for the argument. Note that in all cases we are using Lemma 26, to pose that $\vdash_m (t_2^{G_i}, \nu_i')$. Also in every rule where the starting store is not an evolving store, then we can always apply Lemma 124, to advance an evolving store of the less precise term into a not evolving store in the precision relation.

Case *(RF) and $F[\varepsilon_{12}(\varepsilon_{11} u_1 :: G_{11})] \mid \mu_1 \longmapsto F[\varepsilon_{11}' u_1] \mid \nu_1'$, where $u_1 = o_m^{G_{u1}}$.* By inspection of $(\sqsubseteq_{::})$ $t^{G_2} = F'[\varepsilon_{22}(\varepsilon_{21} u_2 :: G_{21})]$, where $\varepsilon_{11} \sqsubseteq \varepsilon_{21}$, $\varepsilon_{12} \sqsubseteq \varepsilon_{22}$, $G_{11} \sqsubseteq G_{21}$, $u_1 \sqsubseteq u_2$. Therefore by $(\sqsubseteq_o)$, $u_2 = o_m^{G_{u2}}$, for some G_{u2} such that $G_{u1} \sqsubseteq G_{u2}$. If ν_2 is an evolving store then by Lemma 29, $t_1^{G_2} \mid \nu_2 \longmapsto^* t_1^{G_2} \mid \mu_2$, such that $\mu_1 \sqsubseteq \mu_2$.

Suppose $\mu_1(u_1) = \varepsilon_{u1} u_1' :: G_{u1}$ and $\varepsilon_{u1} \not\sqsubseteq \mathit{iref}(\varepsilon_{11}')$, then as $\mu_1 \sqsubseteq \mu_2$, then $\mu_2(u_2) = \varepsilon_{u2} u_2' :: G_{u2}$, where $\varepsilon_{u1} \sqsubseteq \varepsilon_{u2}$, $u_1' \sqsubseteq u_2'$. If $\varepsilon_{11}' = \varepsilon_{11} \circ \varepsilon_{12}$ is defined, and $\varepsilon_{12}' = \mathit{iref}(\varepsilon_{11}') \circ \varepsilon_{u1}$ is defined, then $\nu_1' = \mu_1[u_1 \mapsto \varepsilon_{12}'(\varepsilon_{u1} u_1' :: G_{u1}) :: G_{u1}]$.
By Proposition 64 $\varepsilon_{21}' = \varepsilon_{21} \circ \varepsilon_{22}$ is defined and $\varepsilon_{11}' \sqsubseteq \varepsilon_{21}'$, $\varepsilon_{22}' = \mathit{iref}(\varepsilon_{21}') \circ \varepsilon_{u2}$ is also defined and $\varepsilon_{12}' \sqsubseteq \varepsilon_{22}'$.
Let us first assume that $\varepsilon_{u2} \not\sqsubseteq \mathit{iref}(\varepsilon_{21}')$, then $\nu_2' = \mu_2[u_2 \mapsto \varepsilon_{22}'(\varepsilon_{u2} u_2' :: G_{u2}) :: G_{u2}]$. Then we have to prove that $\mathit{flat}(\nu_1'(u_1)) \sqsubseteq \mathit{flat}(\nu_2'(u_2))$, i.e. $\mathit{iref}(\varepsilon_{11}') \circ \varepsilon_{u1} \sqsubseteq \mathit{iref}(\varepsilon_{21}') \circ \varepsilon_{u2}$, but the result holds by Lemma 64.
If $\neg(\varepsilon_{u2} \not\sqsubseteq \mathit{iref}(\varepsilon_{21}'))$, then we have to prove that $\mathit{iref}(\varepsilon_{11}') \circ \varepsilon_{u1} \sqsubseteq \varepsilon_{u2}$, which holds by Lemma 119.

Case *(Rν) and $t_1^{G_1} \mid \nu_1 \longmapsto t_2^{G_1} \mid \nu_1'$ and $\nu_1' \neq \mu_1'$.* The result holds by Lemma 28.

Case *(Rν) and $t_1^{G_1} \mid \nu_1 \longmapsto t_2^{G_1} \mid \mu_1'$.* The result holds by Lemma 28, and then Lemma 124. □

References

[1] Amal Ahmed, Robert Bruce Findler, Jeremy G. Siek, Philip Wadler, 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, 2011, pp. 201–214.

[2] Amal Ahmed, Dustin Jamner, Jeremy G. Siek, Philip Wadler, Theorems for free for free: parametricity, with and without types, in: ICFP, 2017, pp. 39:1–39:28.

[3] Johannes Bader, Jonathan Aldrich, Éric Tanter, Gradual program verification, in: Işil Dillig, Jens Palsberg (Eds.), Proceedings of the 19th International Conference on Verification, Model Checking, and Abstract Interpretation, VMCAI 2018, in: Lecture Notes in Computer Science, vol. 10747, Springer-Verlag, Los Angeles, CA, USA, 2018, pp. 25–46.

[4] Felipe Bañados Schwerter, Ronald Garcia, Éric Tanter, A theory of gradual effect systems, in: Proceedings of the 19th ACM SIGPLAN Conference on Functional Programming, ICFP 2014, ACM Press, Gothenburg, Sweden, 2014, pp. 283–295.

[5] Felipe Bañados Schwerter, Ronald Garcia, Éric Tanter, Gradual type-and-effect systems, J. Funct. Program. 26 (2016) 19:1–19:69.

[6] Ambrose Bonnaire-Sergeant, Rich Hickey, Typed clojure: an optional type system for clojure, https://typedclojure.org/. (Accessed May 2020).

[7] Gilad Bracha, Pluggable type systems, in: Workshop on Revival of Dynamic Languages, 2004.

[8] Giuseppe Castagna, Victor Lanvin, Gradual typing with union and intersection types, in: ICFP, 2017, pp. 41:1–41:28.

[9] Alonzo Church, A formulation of the simple theory of types, J. Symb. Log. 5 (2) (1940) 56–68.

[10] Matteo Cimini, Jeremy Siek, The gradualizer: a methodology and algorithm for generating gradual type systems, in: Proceedings of the 43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, ACM Press, St Petersburg, FL, USA, 2016, pp. 443–455.

[11] Matteo Cimini, Jeremy G. Siek, Automatically generating the dynamic semantics of gradually typed languages, in: Proceedings of the 44th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2017, ACM Press, Paris, France, 2017, pp. 789–803.

[12] Patrick Cousot, Radhia Cousot, Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, in: Conference Record of the 4th ACM Symposium on Principles of Programming Languages, POPL 77, ACM Press, Los Angeles, CA, USA, 1977, pp. 238–252.

[13] Dart Team, Dart Programming Language Specification, 2013, Version 0.41.

[14] Tim Disney, Cormac Flanagan, Gradual information flow typing, in: International Workshop on Scripts to Programs, 2011.

[15] Facebook Inc., Hack: programming productivity without breaking things, https://hacklang.org/. (Accessed May 2019).

[16] Facebook Inc., Flow is a static type checker for JavaScript, https://flow.org/. (Accessed May 2020).

[17] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, Semantics Engineering with PLT Redex, MIT Press, 2009.

[18] Luminous Fennell, Peter Thiemann, Gradual security typing with references, in: Proceedings of the 26th Computer Security Foundations Symposium, CSF, 2013, pp. 224–239.

[19] Luminous Fennell, Peter Thiemann, LJGS: gradual security types for object-oriented languages, in: Shriram Krishnamurthi, Benjamin S. Lerner (Eds.), Proceedings of the 30th European Conference on Object-Oriented Programming, ECOOP 2016, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 56, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Rome, Italy, 2016, pp. 9:1–9:26.

[20] Ronald Garcia, Alison M. Clark, Éric Tanter, Abstracting gradual typing, in: Proceedings of the 43rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, ACM Press, St. Petersburg, FL, USA, 2016, pp. 429–442.

[21] Ronald Garcia, Éric Tanter, Roger Wolff, Jonathan Aldrich, Foundations of typestate-oriented programming, ACM Trans. Program. Lang. Syst. 36 (4) (2014) 12:1–12:44, Article 12.

[22] Ben Greenman, Matthias Felleisen, A spectrum of type soundness and performance, in: Proceedings of the ACM on Programming Languages, vol. 2, ICFP, Sept. 2018, 2018, pp. 71:1–71:32.

[23] Fritz Henglein, Dynamic typing: syntax and proof theory, Sci. Comput. Program. 22 (3) (1994) 197–230.

[24] David Herman, Aaron Tomb, Cormac Flanagan, Space-efficient gradual typing, Higher-Order Symb. Comput. 23 (2) (2010) 167–189.

[25] William A. Howard, The formulae-as-types notion of construction, in: J.P. Seldin, J.R. Hindley (Eds.), To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism, Academic Press, New York, 1980, pp. 479–490, Reprint of 1969 article.

[26] Atsushi Igarashi, Peter Thiemann, Vasco T. Vasconcelos, Philip Wadler, Gradual session types, in: ICFP, 2017, pp. 38:1–38:28.

[27] Yuu Igarashi, Taro Sekiyama, Atsushi Igarashi, On polymorphic gradual typing, in: ICFP 2017, 2017, pp. 40:1–40:29.

[28] Lintaro Ina, Atsushi Igarashi, 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, 2011, pp. 609–624.

[29] Khurram A. Jafery, Joshua Dunfield, Sums of uncertainty: refinements go gradual, in: Proceedings of the 44th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2017, ACM Press, Paris, France, 2017, pp. 804–817.

[30] Nico Lehmann, Éric Tanter, Gradual refinement types, in: Proceedings of the 44th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2017, ACM Press, Paris, France, 2017, pp. 775–788.

[31] Microsoft Co., Typescript language specification, https://www.typescriptlang.org/. (Accessed June 2017).

[32] Max S. New, Dustin Jamner, Amal Ahmed, Graduality and parametricity: together again for the first time, in: Proceedings of the ACM on Programming Languages, vol. 4, POPL, Jan. 2020, 2020, pp. 46:1–46:32.

[33] Benjamin C. Pierce, Types and Programming Languages, MIT Press, Cambridge, MA, USA, ISBN 0-262-16209-1, 2002.

[34] Ilya Sergey, Dave Clarke, Gradual ownership types, in: Helmut Seidl (Ed.), Proceedings of the 21st European Symposium on Programming Languages and Systems, ESOP 2012, in: Lecture Notes in Computer Science, vol. 7211, Springer-Verlag, Tallinn, Estonia, 2012, pp. 579–599.

[35] Jeremy Siek, Walid Taha, Gradual typing for functional languages, in: Proceedings of the Scheme and Functional Programming Workshop, 2006, pp. 81–92.

[36] Jeremy Siek, Walid Taha, Gradual typing for objects, in: Erik Ernst (Ed.), Proceedings of the 21st European Conference on Object-oriented Programming, ECOOP 2007, in: Lecture Notes in Computer Science, Springer-Verlag, Berlin, Germany, 2007, pp. 2–27.

[37] Jeremy Siek, Philip Wadler, 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, 2010, pp. 365–376.

[38] Jeremy G. Siek, Michael M. Vitousek, Matteo Cimini, John Tang Boyland, Refined criteria for gradual typing, in: 1st Summit on Advances in Programming Languages, SNAPL 2015, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 32, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Asilomar, California, USA, 2015, pp. 274–293.

[39] Jeremy G. Siek, Michael M. Vitousek, Matteo Cimini, Sam Tobin-Hochstadt, Ronald Garcia, Monotonic references for efficient gradual typing, in: Jan Vitek (Ed.), Proceedings of the 24th European Symposium on Programming Languages and Systems, ESOP 2015, in: Lecture Notes in Computer Science, vol. 9032, Springer-Verlag, London, UK, 2015, pp. 432–456.

[40] Peter Thiemann, Luminous Fennell, Gradual typing for annotated type systems, in: Zhong Shao (Ed.), Proceedings of the 23rd European Symposium on Programming Languages and Systems, ESOP 2014, in: Lecture Notes in Computer Science, vol. 8410, Springer-Verlag, Grenoble, France, 2014, pp. 47–66.

[41] Matías Toro, Ronald Garcia, Éric Tanter, Type-driven gradual security with references, ACM Trans. Program. Lang. Syst. 40 (4) (2018) 16:1–16:55.

[42] Matías Toro, Elizabeth Labrada, Éric Tanter, Gradual parametricity, revisited, in: Proceedings of the ACM on Programming Languages, vol. 3, POPL 2019, Jan. 2019, pp. 17:1–17:30.

[43] Matías Toro, Éric Tanter, Customizable gradual polymorphic effects for scala, in: Proceedings of the 30th ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications, OOPSLA 2015, ACM Press, Pittsburgh, PA, USA, 2015, pp. 935–953.
[44] Matías Toro, Éric Tanter, A gradual interpretation of union types, in: Proceedings of the 24th Static Analysis Symposium, SAS 2017, in: Lecture Notes in Computer Science, vol. 10422, Springer-Verlag, New York City, NY, USA, 2017, pp. 382–404.
[45] Niki Vazou, Éric Tanter, David Van Horn, Gradual liquid type inference, in: Proceedings of the ACM on Programming Languages, vol. 2, OOPSLA 2018, Nov. 2018.
[46] Michael M. Vitousek, Andrew M. Kent, Jeremy G. Siek, Jim Baker, Design and evaluation of gradual typing for python, in: Proceedings of the 10th ACM Dynamic Languages Symposium, DLS 2014, ACM Press, Portland, OR, USA, 2014, pp. 45–56, ACM SIGPLAN Notices, 50(2).
[47] Roger Wolff, Ronald Garcia, Éric Tanter, Jonathan Aldrich, Gradual typestate, in: Mira Mezini (Ed.), Proceedings of the 25th European Conference on Object-oriented Programming, ECOOP 2011, in: Lecture Notes in Computer Science, vol. 6813, Springer-Verlag, Lancaster, UK, 2011, pp. 459–483.
[48] Ningning Xie, Xuan Bi, Bruno C.d.S. Oliveira, Consistent subtyping for all, in: Amal Ahmed (Ed.), Proceedings of the 27th European Symposium on Programming Languages and Systems, ESOP 2018, in: Lecture Notes in Computer Science, vol. 10801, Springer-Verlag, Thessaloniki, Greece, 2018, pp. 3–30.