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

CONSULTAS EFICIENTES SOBRE BASES DE DATOS DE GRAFO INCOMPLETAS.

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS, MENCIÓN COMPUTACIÓN.

VÍCTOR ANDRÉS CARMI LARA

PROFESOR GUÍA:
PABLO BARCELÓ BAEZA

PROFESOR COGUÍA:
JORGE PÉREZ ROJAS

MIEMBROS DE LA COMISIÓN:
CLAUDIO GUTIÉRREZ GALLARDO
IVÁN RAPAPORT ZIMERMANN
LORETO BRAVO CELEDÓN

SANTIAGO DE CHILE
2013

RESUMEN DE LA TESIS
PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS, MENCION COMPUTACIÓN
POR: VÍCTOR ANDRES CARMI LARA
FECHA: dd/mm/aaaa
PROF. PABLO BARCELÓ B.
PROF. COGUÍA JORGE PÉREZ R.

CONSULTAS EFICIENTES SOBRE BASES DE DATOS DE GRAFO INCOMPLETAS.

El principal objetivo de esta tesis es encontrar casos tratables y buenas técnicas para computar *Certain Answers* sobre bases de datos de grafos incompletas, en tiempo polinomial.

Las bases de datos de grafos surgen naturalmente de la necesidad de almacenar información de redes, tales como Facebook, mapas de rutas o la web. La idea de "incompletitud" viene de la falta de información completa, complejidad con la cual tenemos que trabajar a diario. Sobre estas bases de datos de grafos carentes de información completa, queremos realizar preguntas en la forma de consultas particulares y determinar si la pregunta puede ser contestada de la misma forma en cada posible completación de la base de datos. La respuesta a estas preguntas en cada posible completación de una base de datos de grafos incompleta es llamada *Certain Answers*.

El problema de *Certain Answers* sobre bases de datos de grafos incompletas ya ha sido estudiado, y es conocido que este problema es en general co-NP completo. En esta tesis convertimos el problema de computar *Certain Answers* en un problema 3-SAT con su fórmula lógica booleana asociada. Podemos probar que bajo ciertas condiciones esta fórmula lógica tiene un número de variables que nos permite determinar si es satisfacible en tiempo polinomial. Luego, probamos que estas condiciones son exhaustivas: sin cualquiera de ellas el problema se vuelve co-NP completo otra vez.

Para analizar más clases tratables del problema de *Certain Answers*, convertimos el problema *Certain Answers* en un problema de programación lineal entera. Con esta formulación de programación lineal, encontramos algunos casos tratables, algoritmos y heurísticas para resolverlo. Sin embargo, el principal logro es la nueva formulación en sí, porque nos permite usar las bien conocidas técnicas de programación lineal para encontrar más casos tratables y mejores heurísticas.

AGRADECIMIENTOS

Agradezco en primer lugar a mi hermano Maurcio que me enseñó desde pequeño a tener gusto por las matemáticas y demás ciencias exactas. A mi madre Jeannette y a mi padre Juan por haber estado siempre dándome apoyo. A Joaquín por haberme respaldado siempre y a mi pequeña y querida hermana Belén por hacerme siempre feliz.

Quiero agradecer luego a mi profesor guía Pablo Barceló por darme la oprtunidad de hacer esta tesis con él y a mi profesor coguía Jorge Pérez por el apoyo que me ha dado. A ambos les quiero agradecer las horas de trabajo en hacer de mí un mejor investigador, las ideas brindadas, las correcciones y la paciencia ante tanta aberración linguística de mi parte.

A mi amiga Alejandra Flores, que cuando asistí a Escuala de Verano me mostró las maravillas que hay en las matemáticas más allá de lo que uno puede ver en el colegio.

A los profesores Claudio Gutiérrez, Iván Rapaport y Loreto Bravo por haber aceptado ser miembros de mi comisión y brindarme importantes correcciones y consejos para ayudar a mejorar este trabajo.

A los académicos Juan Diego Dávila, Alejandro Maass, Marcos Kiwi, Roberto Cominetti y Jaime San Martín por haber hecho cursos exigentes en los cuales logré aprender mucho y adquirir conocimientos que me han servido a lo largo de toda mi carrera.

A Felipe Serrano, quien siempre estuvo apoyandome y dándome ánimo en los momentos difíciles. Los consejos de vida que me ha dado son invaluables.

A Orlando Rivera, con quien he compartido muchas historias buenas y malas, pero siempre ha estado ahí como un gran amigo.

A Italo Cipriano, por haberme acompañado estos años y brindarme tantos momentos felices que siempre recordaré.

A Angélica Zavala, por haberme dado harto tiempo, apoyo y comprensión durante el largo tiempo que hice esta tesis, y así hacer menos pesada esta labor.

A mis amigos, que han estado conmigo en tantos buenos momentos, y apoyándome también en los malos: Francisco Echeverría, Benjamín Obando, Emilio Vilches, Gonzalo Muñoz, Omar Larré, Sergio Araneda, Francisco Unda, Sebastián Donoso, Darío Valdebenito y Miguel Romero.

A mis demás compañeros del DIM y del DCC, con quienes compartimos nuestra formación como estudiantes, y mucho más.

Tabla de contenido

Capítulo 1

Introducción

Realizar consultas y analizar información estructurada en forma de grafos ha recibido mucha antención últimamente, debido a sus numerosas aplicaciones, en áreas tales como redes biológicas [11,12,13], redes sociales [14,15], y la web semántica [16,17]. En estas aplicaciones, la información contenida se modela naturalmente como grafos cuyos nodos son objetos, y las etiquetas de los arcos definen relaciones entre tales objetos [1,18]. Para almacenar este tipo de información usamos bases de datos de grafo, las cuales son grafos etiquetados dirigidos finitos [1,3,4].

El mecanismo más simple para realizar consultas sobre bases de datos de grafo son las *regular path queries* (RPQs) [1,4,5], las cuales consultan la existencia de un camino entre dos nodos que satisfaga ciertas condiciones. De inmediato se observa que las RPQs carecen de poder expresivo en varios aspectos, empezando por el hecho de que no son cerradas bajo conjunción ni menos bajo proyección. Este hecho motiva el uso de las *conjunctive regular path queries* (CRPQs) [1,5,3,6,7], las cuales son conjunciones de RPQs. Esta clase de consultas son para las bases de datos de grafo lo que las *conjunctive queries* [24] son para las bases de datos relacionales. Entre otras cosas, estas consultas permiten expresar patrones complejos de consulta sobre información estructurada como grafos.

La *incompletitud* de información surge en diversos escenarios donde la información está siendo constantemente actualizada, intercambiada o integrada [4]. La idea detrás de la incompletitud es que cierta información puede estar faltando en la base de datos y debe ser reemplazada por valores desconocidos. Una base de datos incompleta puede ser vista como un *patrón*, el cual *representa*, en términos precisos, un (potencialmente infinito) espectro de bases de datos "completas". Más específicamente, todas las bases de datos completas que se pueden obtener reemplazando los valores desconocidos en el patrón de forma consistente. El estudio de la incompletitud se originó para bases de datos relacionales en el artículo seminal de Imielinski y Lipski [22], luego se extendió a XML en el trabajo de Barceló, Libkin, Sirangelo y Poggi [21]. La noción de incompletitud es mucho más compleja en documentos XML [21] que en bases de datos relacionales. En efecto, la información faltante en XML puede aparecer no sólo entre los valores de los atributos, sino que también en la estructura misma.

La incompletitud también ha sido estudiada recientemente en el contexto de bases de

datos de grafo [1]. Las bases de datos de grafo incompletas son, esencialmente, *patrones* de grafo que dan enfásis a la incompletitud a nivel estructural. Más específicamente, los patrones de grafo son grafos dirigidos con arcos etiquetados por expresiones regulares. Un arco etiquetado con una expresión regular r entre nodos a y b representa el hecho de que ambos nodos están conectados por un camino cuyo etiquetado es una palabra que pertenece a la expresión regular r, pero la conexión exacta entre los nodos es desconocida. Luego, un patrón de grafo representa un conjunto de bases de datos de grafo, que son las posibles "completaciones"del patrón. Este conjunto está formado por cada base de datos de grafo que se puede obtener reemplazando cada arco etiquetado con una expresión regular r por un camino cuyo etiquetado forma una palabra perteneciente a la expresión regular r.

En esta tesis se estudia el problema de realizar consultas sobre bases de datos de grafo incompletas. En particular, se concentra en el estudio de las CRPQs. Como los patrones de grafo representan un conjunto potencialmente infinito de bases de datos de grafo "completas", la semántica de las consultas sobre bases de datos incompletas es diferente a la semántica de consultas tradicionales sobre bases de datos. La literatura ha considerado principalmente la semántica de una consulta Q realizada en una base de datos incompleta D sobre la base de las "*certain answers*"[1], las cuales son las respuestas a la consulta Q que están presentes en las respuestas de la consulta Q realizada sobre cada posible completación de D.

Recordemos que la *data complexity* [9] de la evaluación de una consulta es la complejidad medida sólo en términos del tamaño de la información, es decir, suponiendo que la consulta está fija. En esta tesis se estudia la data complexity de computar certain answers de CRPQs sobre bases de datos de grafo incompletas. Es conocido que este problema en general es coNP-completo [1], incluso para RPQs simples. El objetivo de esta tesis es encontrar restricciones naturales para el problema que lleven a casos tratables, así como heurísticas que ayuden a obtener resultados razonables en un escenario general.

Las principales contribuciones de esta tesis son las siguientes:

1. Presentamos una clase natural de bases de datos de grafo incompletas y CRPQs para las cuales se puede solucionar en tiempo polinomial el problema descrito anteriormente. Además, se muestra que la clase tratable es, en un sentido, maximal, debido a que quitando cualquiera de sus condiciones la clase se vuelve intratable; esto es, la complejidad del problema en la clase vuelve a ser coNP-completo.

2. La clase tratable descrita en el punto anterior es, desafortunadamente, un poco restrictiva, y el problema de encontrar clases significativas y menos restrictivas parece ser muy difícil (en particular, en vista de la maximalidad de la clase tratable recién mencionada). Para superar este problema se desarrollan heurísticas basadas en algoritmos para casos generales del problema. En particular, se estudia el caso en que las consultas son CRPQs definidas por expresiones regulares que no usan la estrella de Kleene $*$. La razón es que este caso ya es difícil e interesante, mientras que un escenario más general involucraría grandes dificultades técnicas.

Para diseñar nuestra heurística de aproximación, se muestra que el problema de computar *certain answers* puede ser reducido en forma natural a un problema de programación lineal entera. Esta reducción permite usar técnicas ya bien conocidas de teoría de optimización lineal [2] que funcionan como heurísticas para nuestro problema.

3. Se muestran garantías para las heurísticas, que aseguran correctitud y trabajo en tiempo polinomial para ciertos casos. Uno de estos casos es precisamente la clase tratable que se describió en el punto 1 anterior. Lo que sugiere que las heurísticas presentadas son significativas y robustas, dado que al menos funcionan bien para el caso estudiado en el que el problema se puede resolver en tiempo polinomial. Además, se identifican diferentes casos tratables, los cuales están definidos en función de la estructura de las restricciones del problema de programación lineal entera asociado. Es más, para obtener ejecuciones en tiempo lineal no se necesita chequear la pertenencia a estas clases tratables. Basta con que el problema efectivamente esté en la clase en que se pueda resolver en tiempo polinomial.

4. Finalmente, se estudia una heurística para el caso general, en la cual las CRPQs pueden ser definidas por expresiones regulares que contengan la estrella de Kleene $*$. Esta heurística trabaja en tiempo exponencial en el peor caso. Sin embargo, puede ser detenida antes para obtener una solución aproximada en el sentido de que no hay certeza en el resultado, pero el algoritmo estuvo trabajando y no encontró un testigo de que *certain answers* sea falso, por lo que hay un undicio de que sea verdad.

Esta tesis se estructura en los siguientes capítulos:

2. Estado del Arte. En esta sección se presentan las definiciones básicas que se ocuparán a lo largo de esta tesis. Entre ellas están las nociones de base de datos de grafo, incompletitud y las consultas que serán usadas, las CRPQs.

3. Una solución en tiempo polinomial. Al comienzo de esta sección se formaliza el problema de *Certain Answers* que se estudia en esta tesis. Luego se desarrollan una serie de restricciones que permiten que el problema de computar certain answers pueda resolverse en tiempo polinomial, esto se demuestra en el Teorema 3.6. Luego se muestra que las restricciones mencionadas son exhaustivas puesto que si se elimina cualquiera de ellas, el problema se vuelve coNP-completo. Este resultado se presenta en el Teorema 3.9.

4. Consultas eficientes. En esta sección se presenta una forma alternativa de plantear el problema estudiado, convirtiéndolo en un problema equivalente de programación lineal entera, bajo la condición de que los lenguajes de consulta sean finitos, lo que se demuestra en el Teorema 4.2. Además se presentan algunas heurísticas para resolver el problema planteado en esta nueva forma.

5. Garantías. En esta sección se presentan algunas condiciones bajo las cuales los algoritmos de la sección anterior corran en tiempo polinomial y responden con exactitud. En particular se muestra que las condiciones que aseguran que el problema sea tratable (Teorema 3.6) expuestas en el Capítulo 3 también permiten que estos algoritmos respondan con exactitud en tiempo polinomial.

6. Caso General. En esta sección se levanta la condición de que los lenguajes de consulta sean finitos, y se presenta un algoritmo de heurística que resuelve el problema en este caso. Este algoritmo funciona mediante una representación del problema como un problema de programación lineal entera y ayuda del algoritmo de *generación de columnas* [25].

Capítulo 2

Estado del Arte

En este capítulo se revisará la terminología que se usará a lo largo de este trabajo y se presentarán los principales objetos de estudio.

Bases de Datos de Grafo. Una *base de datos de grafo* es un grafo dirigido finito con etiquetas en sus arcos [1,3,4]. Formalmente, si Σ es un alfabeto finito (un conjunto finito de símbolos), y $\mathcal{N}$ un conjunto infinito numerable de ids de nodos, una base de datos de grafo sobre el alfabeto Σ, de ahora en adelante *GDB* (*graph database*), es un par $G = (N, E)$ donde $N \subseteq \mathcal{N}$ es un conjunto finito de nodos, y E es un conjunto de arcos etiquetados en Σ (es decir, $E \subseteq N \times \Sigma \times N$).

Camino. Sean n_0 y n_m nodos en una base de datos G. Un *camino* ρ desde n_0 hasta n_m es una secuencia

$$(n_0, a_0, n_1), (n_1, a_1, n_2), \cdots, (n_{m-1}, a_{m-1}, n_m),$$

para algún $m > 0$, donde cada (n_i, a_i, n_{i+1}), para $i < m$, es un arco en E. En particular, todos los n_is son nodos en N y todos los a_js son letras en Σ. El etiquetado de ρ, denotado por $\lambda(\rho)$, es la palabra $a_0 \cdots a_{m-1} \in \Sigma^*$. Además, definimos el *camino vacío* como (n, ϵ, n) para cada $n \in N$; el etiquetado de tal camino es la palabra vacía ϵ.

Notación de Expresiones Regulares. Para definir las consultas se usarán lenguajes regulares, los cuales serán expresados a lo largo de esta tesis por expresiones regulares y serán tratados como tales en algunas ocasiones para simplificar notación. Dados lenguajes regulares L_1 y L_2, la notación a usar será:

- Disyunción: Se denotará con | la disyunción, esto es:

$$L_1|L_2 = \{w : w \in L_1 \vee w \in L_2\}.$$

- Concatenación: Se denotará con $\cdot$ la concatenación, o simplemente se juntarán los lenguajes como se hace usualmente para concatenar palabras, esto es:

$$L_1 \cdot L_2 = L_1 L_2 = \{w : w = w_1 w_2, \quad w_1 \in L_1 \wedge w_2 \in L_2\}.$$

- Estrella de Kleene: Se denotará con $*$ a la estrella de Kleene, esto es:

$$L_1^* = \{\epsilon\} \cup L_1 \cup L_1 \cdot L_1 \cup L_1 \cdot L_1 \cdot L_1 \cdots$$

donde ϵ es la palabra vacía. La precedencia de los operadores será $* > \cdot > |$.

Regular Path Queries. El mecanismo más básico de consulta sobre GDBs son las *regular path queries* (*RPQ*) [1,4,5]. Ellas retornan todos los pares de nodos de la base de datos que están conectados por un camino ρ cuyo etiquetado $\lambda(\rho)$ pertenece a un lenguaje regular dado. Formalmente, una RPQ Q es una expresión de la forma $Q = (x, L, y)$, donde $L \subseteq \Sigma^*$ es un lenguaje regular y las variables de nodos x e y no son necesariamente distintas. Dada una base de datos de grafo $G = (N, E)$ y una RPQ Q, ambas sobre Σ, la respuesta $Q(G)$ corresponde al conjunto de todos los pares de nodos $(n_1, n_2) \in N \times N$ que se les pueden asignar como valores a (x, y). Es decir que están conectados por un camino ρ cuyo etiquetado $\lambda(\rho)$ está en L.

Conjunctive Regular Path Queries. Como se vio en la Introducción, es necesario usar consultas más expresivas que las RPQs. Las consultas que se van a usar son las *conjunctive regular path queries* (CRPQ) [1,5,3,6,7]. Una CRPQ Q sobre un alfabeto finito Σ es una expresión de la forma:

$$Ans(\bar{z}) \leftarrow \bigwedge_{1 \leq i \leq m} (x_i, L_i, y_i), \tag{2.1}$$

donde $m > 0$, las (x_i, L_i, y_i) son regular path queries y $\bar{z}$ es una tupla de variables entre las variables de $\bar{x}$ y de $\bar{y}$, recordando que $\bar{x}$ es la notación usual para el vector de los nodos $(x_i)_{i=1}^m$. Una consulta con encabezado $Ans()$ (es decir, sin variables en la salida) es llamada *consulta booleana*. Llamamos $\mathcal{F}_{\text{CRPQ}}$ al conjunto de todas las CRPQs tales que L_i es un lenguaje finito para todo i.

Intuitivamente, una CRPQ Q selecciona tuplas $\bar{z}$ para las cuales existen valores de las demás variables de nodos de $\bar{x}$ y $\bar{y}$ tales que cada RPQ (x_i, L_i, y_i) es satisfecha. Formalmente, dada una CRPQ Q de la forma (2.1) y una GDB $G = (N, E)$, una valuación es un mapeo $\sigma : \bigcup_{1 \leq i \leq m} \{x_i, y_i\} \to N$. Decimos $(G, \sigma) \models Q$ si $(\sigma(x_i), \sigma(y_i))$ está en la respuesta a la RPQ (x_i, L_i, y_i) en G $\forall i = 1, \cdots, m$. Es decir, si existe un camino ρ en G desde $\sigma(x_i)$ hasta $\sigma(y_i)$ etiquetado por $\lambda(\rho_i) \in L_i$. Entonces $Q(G)$ es el conjunto de todas las tuplas $\sigma(\bar{z})$ tales que $(G, \sigma) \models Q$. Si Q es una CRPQ booleana, decimos que $Q(G)$ es verdadero si existe una valuación σ tal que $(G, \sigma) \models Q$, de lo contrario $Q(G)$ es falso. Decimos que las variables en $\bar{z}$ son variables libres y que las demás variables en $\bar{x}$ y en $\bar{y}$ son variables existencialmente cuantificadas. El siguiente ejemplo ilustra este tipo de consultas:

Ejemplo 2.1. Consideremos $G = (N, E)$ una base de datos de grafo con $N = \{n_1, n_2, n_3, n_4, n_5, n_6\}$ y

$$E = \{(n_1, 0, n_2), (n_2, 1, n_3), (n_1, 0, n_3), (n_1, 1, n_4), (n_4, 1, n_5), (n_5, 1, n_6)\}.$$

Esta base de datos de grafo se puede graficar como muestra la Figura 2.2:

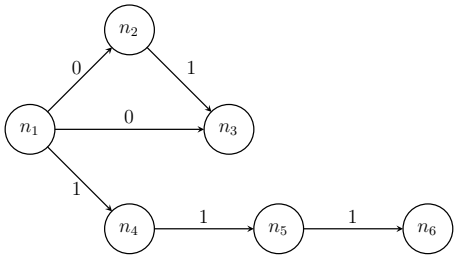

Figura 2.1: Base de datos de grafo G.

Consideremos la consulta Q:

$$Ans(\bar{z}) \leftarrow (x_1, L_1, y_1) \wedge (x_1, L_2, y_2) \wedge (x_1, L_3, y_3)$$

con $\bar{z} = (x_1)$, por tanto x_1 la única variable libre. Luego y_1, y_2 e y_3 son las variables existencialmente cuantificadas, las cuales además se pueden explicitar denotando Q de la siguiente forma:

$$Ans(\bar{z}) \leftarrow \exists y_1, y_2, y_3 (x_1, L_1, y_1) \wedge (x_1, L_2, y_2) \wedge (x_1, L_3, y_3).$$

Los lenguajes L_1, L_2 y L_3 de la consulta Q son los siguientes:

$$L_1 = 0|00|000 \quad L_2 = 01|001|0001 \quad L_3 = 101|111|011$$

Consideremos $\bar{t} = (n_1)$, luego podemos preguntar si es verdad $\bar{t} \in Q(G)$. La respuesta es afirmativa, puesto que podemos considerar σ definido de la siguiente forma:

$$\sigma(x_1) = n_1, \quad \sigma(y_1) = n_2, \quad \sigma(y_2) = n_3, \quad \sigma(y_3) = n_6$$

Notemos que $(\sigma(x_1), \sigma(y_1)) = (n_1, n_2)$ está en la respuesta de (x_1, L_1, y_1) puesto que existe un camino etiquetado por 0 entre n_1 y n_2, donde $0 \in L_1$. De igual manera $(\sigma(x_1), \sigma(y_2)) = (n_1, n_3)$ está en la respuesta de (x_1, L_2, y_2) puesto que existe un camino etiquetado por 01 entre n_1 y n_3, donde $01 \in L_2$. Además $(\sigma(x_1), \sigma(y_3)) = (n_1, n_6)$ está en la respuesta de (x_1, L_3, y_3) puesto que existe un camino etiquetado por 111 entre n_1 y n_6, donde $111 \in L_3$. Finalmente se concluye que $\bar{t} = (n_2) \in Q(G)$. Podemos observar adicionalmente que la función σ que permite comprobar esto no es única, pues podemos considerar $\sigma(y_1) = n_3$.

Bases de Datos de Grafo Incompletas. Se usarán patrones de grafo [1] para representar bases de datos de grafo incompletas. Formalmente, un patrón de grafo π es un grafo (N, E) donde N es un conjunto de nodos constantes, esto es $N \subseteq \mathcal{N}$, y $E \subseteq N \times REG(\Sigma) \times N$, donde $REG(\Sigma)$ es el conjunto de lenguajes regulares sobre Σ. Intuitivamente, un patrón de grafo representa todas las posibles GDBs que se pueden obtener reemplazando los arcos del patrón por caminos cuyo etiquetado es una palabra en el lenguaje regular del arco original del patrón. A cada una de estas GDBs se le llamará *completación* del patrón. Para definir claramente qué GDBs son posibles completaciones de un patrón de grafo se necesita la definición de *homomorfismo*:

Sea $\pi = (N_1, E_1)$ un patrón de grafo y $G = (N_2, E_2)$ una base de datos de grafo. Se dice que $h : N_1 \to N_2$ es un homomorfismo si $h(n) = n$, $\forall n \in N_1$, y para todo arco $(p, L, p') \in E_1$ existe un camino entre $h(p)$ y $h(p')$ en G cuyo etiquetado está en L.

Si $\pi = (N_1, E_1)$ es un patrón de grafo y $G = (N_2, E_2)$ es una GDB, decimos que G es una *completación posible* de π si existe un homomorfismo $h : N_1 \to N_2$. De ahora en adelante, cada vez que se haga referencia a un base de datos de grafo incompleta se hará referencia a un patrón de grafo.

El *espacio de completaciones* de una GDB incompleta π, denotado $Rep_\Sigma(\pi)$, es el espacio de cada completación posible de π con un alfabeto Σ. Formalmente para $\pi = (N_1, E_1)$:

$$Rep_\Sigma(\pi) = \{G = (N_2, E_2) \text{ una GDB} | \text{ existe un homomorfismo } h : N_1 \to N_2\}.$$

Ejemplo 2.2. Consideremos $\pi = (N', E')$ una base de datos de grafo incompleta con $N' = \{n_1, n_3, n_6\}$ y

$$E' = \{(n_1, 0^*1, n_3), (n_1, 0, n_3), (n_5, 1^*, n_6)\}.$$

Esta base de datos de grafo se puede graficar como muestra la Figura 2.2:

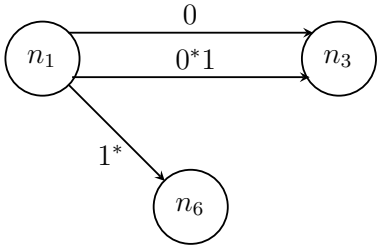

Figura 2.2: Base de datos de grafo incompleta π.

Notemos que la base de datos G del Ejemplo 2.1 es una completación de π. Para verificar esto basta considerar el homomorfismo $h : N \to N'$ tal que:

$$h(n_1) = n_1, \quad h(n_3) = n_3 \quad h(n_6) = n_6$$

donde se cumple que en G entre los nodos n_1 y n_3 hay un camino etiquetado por la palabra 01, la cual pertenece al lenguaje regular asociado al arco $(n_1, 0^*1, n_3)$ de π. Entre los nodos

n_1 y n_3 hay un camino etiquetado por la palabra 0, la cual pertenece al lenguaje regular asociado al arco $(n_1, 0, n_3)$ de π. Finalmente, los nodos n_1 y n_6 hay un camino etiquetado por la palabra 111, la cual pertenece al lenguaje regular asociado al arco $(n_5, 1^*, n_6)$ de π. Con esto verificamos que G es una completación de π.

Grafo Completo Inducido. Dada una base de datos de grafo incompleta π, el *grafo completo inducido* por π, denotado como $comp(\pi)$, es la base de datos de grafo obtenida de π removiendo cada arco incompleto de π, esto es, cada arco etiquetado con una expresión regular que es sintácticamente distinta de un símbolo en el alfabeto Σ asociado al patrón.

Ejemplo 2.3. Para la base de datos incompleta π del Ejemplo 2.2, el grafo completo inducido sería $G_c = (N_c, E_c)$ con $N_c = \{n_1, n_3, n_6\}$ y $E_c = \{(n_1, 0, n_3)\}$.

Certain Answers. Consideremos consultas Q que reciben bases de datos de grafo como entrada y retornan conjuntos de tuplas de sus nodos. Por ejemplo, RPQs y CRPQs son consultas como las descritas. Para ellas, se puede definir sus *certain answers*, denotadas $\Box Q(\pi)$, sobre patrones de grafos π de la forma estándar:

$$\Box Q(\pi) = \bigcap_{G \in Rep(\pi)} Q(G) \text{ [22].}$$

En el caso booleano la respuesta es verdadero ssi $Q(G)$ es verdadero $\forall G \in Rep(\pi)$.

Ejemplo 2.4. Consideremos la consulta Q del Ejemplo 2.1 y el patrón π del Ejemplo 2.2. Como se vió en el Ejemplo 2.1 sabemos que existe una base de datos de grafo G con $(n_1) \in Q(G)$. Revisando los demás nodos de G se puede comprobar que la consulta no puede ser evaluada tomando como nodo libre (x_1) ninguno de los nodos restantes de G, luego $Q(G) = \{(n_1)\}$.

Consideremos la base de datos de grafo $\bar{G} = (\bar{N}, \bar{E})$ (ver Figura 2.3) con $\bar{N} = \{n_1, n_3, n_6\}$ y $\bar{E} = \{(n_1, 1, n_3), (n_1, 0, n_3), (n_5, 1, n_6)\}$. La cual es una completación de π, lo que se puede probar utilizando el mismo homorfismo que en el Ejemplo 2.2.

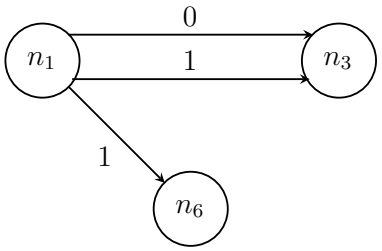

Figura 2.3: Base de datos de grafo $\bar{G}$.

Si tomamos n_1 como el nodo libre en la consulta Q, podemos observar que no es posible evaluar el tercer término de de la consulta (x_1, L_3, y_3), donde $L_3 = 101|111|011$. Esto ocurre debido a que no hay caminos de largo mayor a 1, y para evaluar la consulta necesitamos un camino que empieze en n_1 cuyo etiquetado esté en $L_3 = 101|111|011$. Por lo tanto $(n_1) \notin Q(\bar{G})$.

Como $G, \bar{G} \in Rep(\pi)$ y se tiene $(n_1) \notin Q(\bar{G})$ y $Q(G) = \{(n_1)\}$, obtenemos que $\Box Q(\pi) = \phi$.

Capítulo 3

Una solución en tiempo polinomial

En este capítulo se define formalmente el problema de *Certain Answers* se va a estudiar. En la primera sección se ilustran algunas características que hacen que el problema que se aborda en esta tesis sea coNP-completo y el Teorema 3.6, que muestra que restringiendo las características ilustradas se puede resolver el problema de *Certain Answers* en tiempo polinomial. En la segunda sección se muestra que el Teorema 3.6 es exhaustivo, probando que si no se tiene cualquiera de las condiciones requeridas el problema se vuelve coNP-completo. Esto se demuestra en el Teorema 3.9.

El problema que se estudiará en este trabajo es *data complexity* [9] de computar Certain Answers en GDBs incompletas, es decir, la complejidad del problema de responder consultas cuando la consulta está fija. Esta es una suposición usual en teoría de bases de datos puesto que las consultas suelen ser más pequeñas que la información almacenada en la base de datos. Formalmente, se estudiará el siguiente problema:

PROBLEMA	DATA COMPLEXITY(Q)
ENTRADA	UN PATRÓN $\pi = (N, E)$, UNA TUPLA $\bar{v} \in N^k$.
PREGUNTA	¿ES CIERTO QUE $\bar{v} \in \Box Q(\pi)$?

(3.1)

La complejidad de este tipo de consultas es elevada en la mayoría de los casos, pues su clase de complejidad computacional es coNP-completo. El siguiente teorema fue enunciado en el trabajo de Barceló, Libkin y Reutter [1].

Teorema 3.1. *[1] El problema de data complexity de responder CRPQs sobre clases de patrones de grafo es coNP-completo, incluso si la consulta es una RPQ $Q = (x, w, y)$, donde $w \in \Sigma^*$ es una palabra.*

El resultado previo indica que incluso si la consulta es muy simple, si no imponemos condiciones especiales sobre la base de datos de grafo incompleta o sobre la consulta, el problema no es tratable. En este capítulo se buscan condiciones que permitan resolver el problema eficientemente. Nuestra clase de patrones se define mediante dos condiciones, que

juntas brindan tratabilidad al problema de computar *Certain Answers* para una clase de CRPQs con ciertas características (las cuales se mencionarán más adelante). Más adelante se mostrará que que la clase de patrones obtenida es, en un sentido preciso, maximal para tratabilidad.

3.1 Una solución en tiempo polinomial

Evaluar Certain Answers en general es muy difícil. Se restringirá el problema a una clase de patrones y consultas, donde es posible computar Certain Answers en tiempo polinomial con respecto al tamaño de la base de datos sobre la que se consulta. La primera restricción es sobre el grado de salida de los nodos en la GDB incompleta. Mientras más grande sea el grado de salida de un nodo, mayor es el número de caminos que se pueden formar que salen de ese nodo. Luego para revisar una consulta es necesario revisar un mayor número de caminos. Esto motiva la primera restricción que se va a estudiar, para la cual es necesaria la siguiente definición:

Definición 3.2. Dada una GDB incompleta π, llamaremos *grado de salida de* π al máximo grado de salida de sus nodos.

Se restringe el problema a la clase de GDBs incompletas cuyo grado de salida (esto es, el máximo grado de salida de sus nodos) es acotado por una constante d. Esta clase se define formalmente de la siguiente forma:

Definición 3.3. Dada una GDB incompleta π, y $d \geq 0$ una constante dada, entonces llamaremos $\mathcal{OD}_{\leq d}$ al conjunto de GDBs incompletas π tal que su grado de salida es acotado por d.

Restringir el problema de computar Certain Answers a la clase de GDBs incompletas $\mathcal{OD}_{\leq d}$ no es suficiente para poder resolverlo en tiempo polinomial porque la complejidad del problema sigue siendo coNP-completo. Este hecho se prueba más adelante como un corolario directo del Teorema 3.9.

La siguiente restricción es introducida por el siguiente ejemplo.

Ejemplo 3.1. Sea n impar, se define π_n como sigue:

Figura 3.1: π_n

Consideremos la CRPQs $Q = \exists x \exists y(x, aa|bb, y)$ y $Q' = \exists x \exists y(x, (a|b)bc, y)$. Notemos que las Certain Answers de Q y Q' sobre π_n son verdaderas, lo que se explica a continuación. Para el caso de la evaluación de Q, sea cual sea la base de datos que se considere en $Rep(\pi_n)$,

ésta tiene que tener etiquetas a o b en los arcos entre v_i y v_{i+1} para $1 \leq i \leq n-2$, luego si hay dos arcos seguidos con a o con b la consulta se evalua afirmativamente, por lo que la única forma de no poder evaluar la consulta es que las etiquetas de los arcos vayan alternando a y b, empezando por b y terminando con a. Esto no es posible puesto que hay $n-2$ arcos etiquetados con el lenguaje $a|b$ en π_n, luego como n es impar hay una cantidad impar de estos arcos, así se tiene que para cualquier base de datos en $Rep(\pi_n)$ es posible evaluar Q. Para el caso de la evaluación de Q' como la consulta tiene una c en su lenguaje, basta inspeccionar los tres últimos nodos de π_n, dónde se verifica fácilmente que para cualquier base de datos en $Rep(\pi_n)$ es posible evaluar Q'. □

Notemos que la dificultad para computar *Certain Answers* de Q en π_n es que es necesario revisar un mayor número de nodos que los que son necesarios de revisar con Q'. Estos nodos que hay que revisar están asociados directamente a los nodos que son posibles de asignar a la consulta para que sea evaluada en alguna completación del patrón. Es decir, si hay más nodos que se pueden utilizar para evaluar la consulta en alguna completación del patrón, habrá un mayor número de nodos por revisar para verificar *Certain Answers*.

Para formalizar la intuición que brinda el ejemplo anterior, mediante funciones que indican los nodos que se pueden utilizar en alguna completación para evaluar la consulta, se tiene la siguiente definición:

Definición 3.4. Sea $\pi = (N, D)$ un patrón de grafo y Q una CRPQ. Sea $\bar{x}$ la tupla de variables libres en Q, $\bar{y}$ la tupla de variables existencialmente cuantificadas en Q, y consideremos una función $f : \bar{y} \to N \cup D$. Dada una tupla $\bar{t}$ decimos que f es una *meaningful function* para π, Q y $\bar{t}$ si existe una base de datos de grafo $G \in Rep(\pi)$ y un mapeo σ tal que:

1. G se obtiene de π reemplazando cada arco de la forma $(n, r, m) \in D$ por un camino ρ_e, de nuevos ids de nodo, etiquetado con una palabra w en el lenguaje regular r.

2. El mapeo σ es un mapeo de las variables de Q en los nodos de G, que verifica $\bar{t} \in Q(G)$, en particular, $\sigma(\bar{x}) = \bar{t}$.

3. Para todo y en $\bar{y}$ se tiene

$$\sigma(y) = \begin{cases} f(y) & \text{si } f(y) \in N \\ n & \text{si } f(y) \in D \text{ y } n \text{ es un nodo en } \rho_{f(y)} \end{cases}$$

Llamaremos $mf(Q, \pi, \bar{t})$ a la cantidad de *meaningful functions*.

Ejemplo 3.2. Consideremos la GDB incompleta $\pi = (N', E')$ del Ejemplo 2.2, la CRPQ Q del Ejemplo 2.1 y la tupla $\bar{t} = (n_1)$. Sea $\bar{x} = (x_1)$ la tupla de variables libres en Q e $\bar{y} = (y_1, y_2, y_3)$ la tupla de variables existencialmente cuantificadas en Q.

Sea $f : \bar{y} \to N' \cup E'$ definida de la siguiente forma:

$$f(y_1) = (n_1, 0^*1, n_3), \quad f(y_2) = n_3, \quad f(y_3) = n_6,$$

Como vimos en el Ejemplo 2.2, se puede comprobar que existe una base de datos de grafo $G = (N, E) \in Rep(\pi)$ con $N = \{n_1, n_2, n_3, n_4, n_5, n_6\}$ y

$$E = \{(n_1, 0, n_2), (n_2, 1, n_3), (n_1, 0, n_3), (n_1, 1, n_4), (n_4, 1, n_5), (n_5, 1, n_6)\},$$

la cual nos permite satisfacer la primera condición para que f sea una *meaningful function*.

Considerando la misma función σ del Ejemplo 2.1 que permite comprobar que $\bar{t} \in Q(G)$, definida de la siguiente forma:

$$\sigma(x_1) = n_1, \quad \sigma(y_1) = n_2, \quad \sigma(y_2) = n_3, \quad \sigma(y_3) = n_6,$$

podemos verificar que se tiene la segunda condición.

Finalmente tenemos que $\sigma(y_1) = n_2$ que es un nodo en el camino $\rho_{f(y_1)}$, el cual es el camino en G que proviene del arco $(n_1, 0^*1, n_3)$ en π. Además, $\sigma(y_2) = n_3 = f(y_2)$ y $\sigma(y_3) = n_6 = f(y_3)$, lo que prueba la tercera condición. Por lo tanto f es una *meaningful function*.

Es importante notar que la idea detrás de las *meaningful functions* es, en palabras simples, que tratan de asemejarse a las funciones σ que indican como evaluar la consulta en una base de datos de grafo. Salvo que como las *meaningful functions* están asociadas a bases de datos de grafo incompletas pueden indicar arcos además de nodos, puesto que al obtener una completación del patrón, el arco será reemplazado por un camino de nodos.

Supongamos que una consulta Q no se puede evaluar directamente en la información completa de π ($\bar{t} \notin Q(comp(\pi))$). Luego uno puede esperar que la interacción entre π y Q sea más bien sofisticada, puesto que es necesario estudiar los arcos incompeltos de π. Por lo tanto Q debería tener solamente una pequeña cota en el número de potenciales nodos sobre los cuales se podría verificar en una completación del patrón π, para poder computar *Certain Answers* de manera rápida. Esto es precisamente lo que la siguiente definición expresa, suponiendo una cota logarítmica, puesto que las meaningful functions esencialmente codifican los potenciales nodos en que se podría evaluar Q en una completación del patrón.

Definición 3.5. Dado $k \geq 0$, una CRPQ Q y a tupla $\bar{t}$. Llamamos $\mathcal{MF}^{Q,\bar{t}}_{\leq k}$ a la clase de patrones de grafo π para los cuales se tiene $mf(Q, \pi, \bar{t}) \leq k \log(|\pi|)$, o se tiene que π satisface $\bar{t} \in Q(comp(\pi))$.

Desafortunadamente, incluso para patrones en la clase $\mathcal{OD}_{\leq d} \cap \mathcal{MF}^{Q,\bar{t}}_{\leq k}$, se puede dar el caso que la complejidad sea coNP-completo. Este hecho se demostrará más adelante como un corolario directo del Teorema 3.9. Sin embargo, si se consultan solamente CRPQs en $\mathcal{F}_{\text{CRPQ}}$ (el conjunto de CRPQs $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ tales que L_i es finito para todo i), entonces se puede acotar el número de distintos etiquetados posibles en los arcos del patrón. Juntando las 3 restricciones ($\pi \in \mathcal{OD}_{\leq d}$, $\pi \in \mathcal{MF}^{Q,\bar{t}}_{\leq k}$ y $Q \in \mathcal{F}_{\text{CRPQ}}$), se puede computar Certain Answers en tiempo polinomial como se establece en el siguiente teorema.

Teorema 3.6. *Sea $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ una CRPQ en $\mathcal{F}_{CRPQ}$, $\bar{t}$ una tupla de ids de nodo de la misma aridad que Q, y $d, k \geq 0$ valores fijos. Entonces existe un algoritmo que, dado un patrón π tal que se tiene $\pi \in \mathcal{OD}_{\leq d} \cap \mathcal{MF}_{\leq k}^{Q,\bar{t}}$ o se tiene $\bar{t} \in Q(comp(\pi))$, verifica $\bar{t} \in \Box Q(\pi)$ en tiempo polinomial en el tamaño de π.*

Demostración. Esta demostración se basa en encontrar una fórmula 3CNF, donde una asignación para esta fórmula que la haga satisfacible se corresponde con una asignación de etiquetas en π que hacen imposible de evaluar la consulta. La fórmula se construye en términos en términos de particiones de palabras y de las *meaningful functions*, de forma de encerrar todas las posibles formas de evaluar la consulta. Luego se prueba la equivalencia expresando el significado de que se satisfaga la fórmula, en la correspondiente asignación de etiquetas para π.

Notemos que si tenemos $\bar{t} \in Q(comp(\pi))$, es obvio que se puede evaluar la consulta en cada posible completación del patrón π, y es una condición fácil de verificar. Para verificar si $\bar{t} \in Q(comp(\pi))$ sólo se necesita remover en π cada arco incompleto, y luego evaluar la consulta en la resultante GDB.

Sea $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ una CRPQ donde cada L_i es un lenguaje finito, $\bar{t}$ una tupla de ids de nodos de la misma aridad que Q, π un patrón de grafos en $\mathcal{OD}_{\leq d} \cap \mathcal{MF}_{\leq k}^{Q,\bar{t}}$ y $(f_j)_{j=1}^{mf(Q,\pi,\bar{t})}$ las *meaningful functions* para Q, π y $\bar{t}$, pero extendidas a todas las variables en $Q(\bar{t})$ mediante $f_j(\bar{t}) = \bar{t}$, $\forall j$.

Se probará el teorema mediante la creación de una fórmula lógica φ de largo logarítmico en el tamaño de π tal que φ es satisfacible si y sólo si $\bar{t} \notin \Box Q(\pi)$.

Primero, es necesario definir algunos conjuntos:

- El conjunto de particiones de la palabra w, $\mathcal{P}_w = \{(w_l)_{l=1}^{k} | w_1 \dots w_k = w, k \leq |w|\}$.
- El conjunto de caminos de arcos que van desde $f_j(x_i)$ a $f_j(y_i)$,

$$C_{j,k,i} = \{(e_1, \cdots, e_k)\text{camino en } \pi | \ (f_j(x_i) = t(e_1) \vee f_j(x_i) = e_1) \\ \wedge (f_j(y_i) = h(e_k) \vee f_j(y_i) = e_k)\}.$$

- Un conjunto técnico, el conjunto de tuplas de particiones de palabras de las tuplas de palabras en los lenguajes asociados a las conjunciones de la consulta Q,

$$\mathcal{P}_{\mathcal{Q}} = \bigcup_{(w_1,\dots,w_m) \in L_1 \times \cdots \times L_m} \mathcal{P}_{w_1} \times \cdots \times \mathcal{P}_{w_m}.$$

Los literales en la fórmula son las expresiones $L_e = \omega$, $L_e = \omega\Sigma^*$, $L_e = \Sigma^*\omega$ y $L_e = \Sigma^*\omega\Sigma^*$ donde L_e es el lenguaje regular en el arco e en π y ω es alguna palabra.

La siguiente fórmula φ_A es la parte principal de la fórmula que queremos construir. Intuitivamente para que φ_A sea verdadera se necesita que para toda meaningful function f_j, para toda forma de escoger palabras $(w_1, ..., w_m)$ en los lenguajes L_i de la consulta Q y para toda forma de particionar esas palabras en $(\bar{w}_1, ..., \bar{w}_m)$, siempre existe algún índice i entre 1 y m, tal que para todo camino c en π entre los nodos $f_j(x_i)$ y $f_j(y_i)$ de largo k_i, donde k_i es la cantidad de palabras en la partición $\bar{w}_i$, se tiene que en el camino c no se puede evaluar la palabra w_i particionada como $\bar{w}_i$.

Formalmente, sea φ_A la siguiente fórmula:

$$\varphi_A = \bigwedge_{j=1}^{mf(Q,\pi,\bar{t})} \bigwedge_{(\bar{w}_1,...,\bar{w}_m)\in\mathcal{P}_\mathcal{Q}} \bigvee_{i=1}^{m} \bigwedge_{c=(e_1,...,e_{k_i})\in C_{j,k_i,i}} \varphi_{c,j,i,\bar{w}_i}$$

donde $\bar{w}_i = (w_{i,l})_{l=1}^{k_i}$, y $\varphi_{c,j,i,(w_l)_{l=1}^k}$ toma diferentes valores acorde al camino $c, j, i, (w_l)_{l=1}^k$:

1. Si $c = (e)$ y $f_j(x_i) = e = f_j(y_i)$, entonces $\varphi_{c,j,i,(w_l)_{l=1}^k}$ es la fórmula:

$$\neg L_e = \Sigma^* \omega \Sigma^*.$$

2. Si $f_j(x_i) = t(e_1)$ y $f_j(y_i) = e_k$, entonces $\varphi_{c,j,i,(w_l)_{l=1}^k}$ es la fórmula:

$$\bigvee_{l=1}^{k-1} \neg L_{e_l} = w_l \vee \neg L_{e_k} = w_k \Sigma^*.$$

3. Si $f_j(x_i) = e_1$ y $f_j(y_i) = h(e_k)$, entonces $\varphi_{c,j,i,(w_l)_{l=1}^k}$ es la fórmula:

$$\neg L_{e_1} = \Sigma^* w_1 \vee \bigvee_{l=2}^{k} \neg L_{e_l} = w_l.$$

4. Si $f_j(x_i) = e_1$ y $f_j(y_i) = e_k$, entonces $\varphi_{c,j,i,(w_l)_{l=1}^k}$ es la fórmula:

$$\neg L_{e_1} = \Sigma^* w_1 \vee \bigvee_{l=2}^{k-1} \neg L_{e_l} = w_l \vee \neg L_{e_k} = w_k \Sigma^*.$$

5. Si $f_j(x_i) = t(e_1)$ y $f_j(y_i) = h(e_k)$, entonces $\varphi_{c,j,i,(w_l)_{l=1}^k}$ es la fórmula:

$$\bigvee_{l=1}^{k} \neg L_{e_l} = w_l.$$

La fórmula φ_A no es suficiente por sí sola, es necesario imponer en la fórmula el hecho de que un arco puede tomar un solo valor a la vez y otras condiciones técnicas con respecto que si en un arco se tiene un etiquetado ω también se tienen de cierta forma los etiquetados de las subpalabras de ω. Para completar la fórmula necesitamos definir $\varphi_B, \varphi_C, \varphi_D$ y φ_E:

$$\begin{aligned}
\varphi_B &= \bigwedge_{e\in E}\left(\bigvee_{\omega\in L_e}(L_e=\omega)\wedge\bigwedge_{\omega\in L_e}\bigwedge_{\omega'\in L_e\setminus\{\omega\}}(\neg(L_e=\omega)\vee(L_e=\omega'))\right),\\
\varphi_C &= \bigwedge_{e\in E}\bigwedge_{\omega\in L_e}\bigwedge_{\alpha\in PF(\omega)}(\neg(L_e=\omega)\vee(L_e=\alpha\Sigma^*)),\\
\varphi_D &= \bigwedge_{e\in E}\bigwedge_{\omega\in L_e}\bigwedge_{\alpha\in SF(\omega)}(\neg(L_e=\omega)\vee(L_e=\Sigma^*\alpha)),\\
\varphi_E &= \bigwedge_{e\in E}\bigwedge_{\omega\in L_e}\bigwedge_{\alpha\in SS(\omega)}(\neg(L_e=\omega)\vee(L_e=\Sigma^*\alpha\Sigma^*)).
\end{aligned}$$

Donde $SF(\omega)$ es el conjunto de sufijos de ω, $PF(\omega)$ es el conjunto de prefijos de ω y $SS(\omega)$ es el conjunto de subpalabras de ω. Luego, la fórmula que buscamos es:

$$\varphi = \varphi_A \wedge \varphi_B \wedge \varphi_C \wedge \varphi_D \wedge \varphi_E.$$

Para construir la fórmula φ es necesario encontrar todas las *meaningful functions*, debido a que φ_A las necesita. El siguiente lema muestra como verificar cuando una GDB incompleta π satisface la hipótesis $\pi \in \mathcal{OD}_{\leq d} \cap \mathcal{MF}^{Q,\bar{t}}_{\leq k}$, y en ese caso el lema muestra como encontrar las *meaningful functions* en tiempo polinomial.

Lema 3.7. *Dada una CRPQ Q, una tupla $\bar{t}$ de la misma aridad que Q, y valores $d, k \geq 0$, verificar si un patrón π pertenece a $\mathcal{OD}_{\leq d} \cap \mathcal{MF}^{Q,\bar{t}}_{\leq k}$ se puede hacer en tiempo polinomial en el tamaño de π. Además si π pertenece a la clase se pueden encontrar las meaningful functions asociadas en tiempo polinomial.*

Demostración. Dada una tupla $\bar{t}$, una CRPQ $Q = \wedge_{i=1}^{m}(x_i, L_i, y_i)$, valores $d, k \geq 0$ y un patrón de grafos $\pi = (N, D)$ con $d_{out}(\pi) \leq d$ hay que verificar si se tiene $\bar{t} \in Q(comp(\pi)) \vee mf(Q, \pi, \bar{t}) \leq k \log |\pi|$ en tiempo polinomial. Es fácil ver que se puede verificar $\bar{t} \in Q(comp(\pi))$ en tiempo polinomial puesto que se pueden tomar todas las combinaciones ids de nodos posibles para las variables x_i e y_i de Q existencialmente cuantificadas para extender $\bar{t}$ a t^*, el cual fija todas las variables de nodo de Q en nodos de $comp(\pi)$. Luego, para cada (x_i, L_i, y_i) en Q x_i e y_i están fijos. Se puede crear un autómata con el grafo definido por $comp(\pi)$, cuyo nodo de inicio es x_i y su nodo de termino es y_i, y luego intersectar el lenguaje del autómata resultante con el lenguaje L_i. Si existe una combinación de nodos tal que para todo i no hay intersección vacía, entonces $\bar{t} \in Q(comp(\pi))$. El número de combinaciones está acotado por $\mathcal{O}(|\pi|^{2m})$, luego se puede hacer la verificación en tiempo polinomial.

Para verificar $mf(Q, \pi, \bar{t}) \leq k \log |\pi|$ también se extiende $\bar{t}$ a t^* haciendo las combinaciones de ids de nodos sobre las variables de nodo existencialmente cuantificadas en Q, pero esta

vez podemos asociar las variables de nodo a arcos en π. Si asociamos un arco a una variable de nodo x_i se usará como nodo de referencia al nodo que es la cola del arco, si asociamos un arco a una variable de nodo y_i se usará como nodo de referencia al nodo que es la cabeza del arco. Por lo tanto se puede usar t^* como una tupla de nodos, usando los nodos de referencia en cada respectiva combinación de arcos y nodos para las variables existencialmente cuantificadas de Q. El número de combinaciones es menor que $(|N|+|D|)^{2m}$, lo cual es polinomial.

Sea $K = \text{máx}_{w \in \cup_{i=1}^{m} L_i} |w|$, entonces para cada combinación de nodos y arcos obtenemos un t^*, luego para cada $i \leq m$ se toman los caminos de largo a lo más K en π de x_i a y_i y se crea el patrón de grafos G_{i,t^*} con los nodos y arcos de esos caminos. Sea $G = (N', D')$ el patrón de grafos construido con la unión de los patrones G_{i,t^*}. Como se tiene acotado $d_{out}(\pi)$, el número de arcos usados en cada G_{i,t^*} es a lo más $\sum_{j=1}^{K} d_{out}(\pi)^j$ y el tamaño de G es a lo más $m \log |\pi| \sum_{j=1}^{K} d_{out}(\pi)^j$. Entonces se tiene un patrón de grafos G con cada nodo involucrado en la evaluación de la consulta Q usando t^*, y su tamaño es $\mathcal{O}(\log |\pi|)$.

Ahora, para completar el lema 3.7 se necesita la siguiente proposición:

Proposición 3.8. *Sea π una GDB incompleta, y $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ una CRPQ en $\mathcal{F}_{CRPQ}$ (L_i es finito $\forall i$), entonces existe una base de datos incompleta π' en la cual cada arco incompleto e está etiquetado con un lenguaje finito L_e, tal que:*

$$t \in \Box Q(\pi) \Leftrightarrow t \in \Box Q(\pi'), \quad \forall t$$

Esta GDB incompleta puede ser construida en tiempo polinomial en el tamaño de π.

Para el lector interesado, la demostración de la Proposición 3.8 está en el Apéndice.

Gracias a la Proposición 3.8 se puede suponer que los lenguajes regulares en los arcos de π son todos finitos, y por lo tanto los lenguajes en los arcos de G también lo son. Ahora se tiene una GDB incompleta G con $\mathcal{O}(\log |\pi|)$ número de arcos con lenguajes finitos en ellos. Estos lenguajes contienen palabras de a lo más largo polinomial en el tamaño de π. Por lo tanto se tiene a lo más una cantidad polinomial de GDBs en $Rep(G)$ de tamaño polinomial.

Se puede tomar cada completación $G' \in Rep(G)$, y cada mapeo σ de las variables de la consulta Q a los nodos de la GDB G', notando que hay una cantidad polinomial de estos mapeos en el tamaño de G. Los mapeos σ que permiten evaluar Q en G' indican cuales son las meaningful functions de la siguiente forma: Si σ permite evaluar la consulta Q en G', entonces para toda variable x en la consulta Q tal que $\sigma(x) = n$ si el nodo n no es un nodo en π ($n \notin N$), se reasigna $\sigma(x)$ al arco e de π de donde proviene n, esto es el arco e de π que también está en G tal que al obtener G' desde G se reemplaza el arco e por un camino que contiene al nodo n. Por lo tanto el mapeo resultante σ es una meaningful function. Este proceso tarda tiempo polinomial para cada base de datos en $Rep(\pi)$. Esto completa la demostración del lema 3.7. □

Ahora continuamos con la demostración del Teorema 3.6. A continuación se probará que φ es satisfacible si y sólo si $\bar{t} \notin \Box Q(\pi)$.

Si φ es satisfacible, entonces existe una valuación para las variables que hace que φ_A, φ_B, φ_C, φ_D y φ_E sean verdaderas. Puesto que φ_B es verdadera, la valuación hace una asignación a los lenguajes en cada arco $e \in E$. Como φ_B es verdadera hay sólo una variable $L_e = \omega$ que es verdadera para cada arco $e \in E$, luego se puede crear una asignación de palabras para cada arco en π usando las palabras ω que están en las variables $L_e = \omega$ que son verdaderas. Con esta asignación de palabras a los arcos se puede construir una base de datos de grafo $G \in Rep(\pi)$. Las formulas φ_C, φ_D y φ_E aseguran que las variables $L_e = \omega\Sigma^*$, $L_e = \Sigma^*\omega$ y $L_e = \Sigma^*\omega\Sigma^*$ tienen el valor correspondiente a las variables $L_e = \omega$, esto es, si el arco e está etiquetado con la palabra ω en G y si ω' es un prefijo (sufijo) de ω entonces la variable $L_e = \omega'\Sigma^*$ ($L_e = \Sigma^*\omega'$) es verdadera, y si ω' es subpalabra de ω entonces $L_e = \Sigma^*\omega'\Sigma^*$ también es verdadera.

La fórmula φ_A asegura que con la asignación de palabras, para toda meaningful function f_j, toda asignación de palabras $(w_1, ..., w_m)$ en la consulta con cada forma de particionarlas en $(\bar{w}_1, ..., \bar{w}_m)$, existe al menos un i tal que $1 \leq i \leq m$ y para todo camino $c = (e_1, ..., e_{k_i})$ entre $f_j(x_i)$ y $f_j(y_i)$ se tiene que $\varphi_{c,j,i,\bar{w}_i}$ es verdadera. Lo que significa que hay en el camino c al menos un arco e_l con etiquetado en G que no está etiquetado con $w_{i,l}$, por lo tanto no se puede evaluar (x_i, w_i, y_i), con w_i particionado como $\bar{w}_i = (w_{i,l})_{l=1}^{k_i}$ en el camino c en G.

Supongamos que $\bar{t} \in Q(G)$, luego existe una meaningful function f tal que indica en que nodos (o arcos) se puede evaluar la consulta, una asignación $(w_1, ..., w_m)$ de palabras para la consulta, una forma de particionarlas en $(\bar{w}_1, ..., \bar{w}_m)$ tal que para todo i entre 1 y m existe un camino $c = (e_1, ..., e_{k_i})$ en el cual (x_i, w_i, y_i) puede ser evaluado con w_i particionado como $\bar{w}_i$, pero esto es una contradicción con lo establecido en el párrafo anterior. Entonces $\bar{t} \notin Q(G)$, y por lo tanto $\bar{t} \notin \Box Q(\pi)$.

Si $\bar{t} \notin \Box Q(\pi)$ hay una base de datos de grafo $G \in Rep(\pi)$ tal que $\bar{t} \notin Q(G)$. Se puede usar la asignación de arcos para π que permite construir G, para asignar valores a los literales en φ. En efecto, para cada arco e, si está etiquetado por w en G entonces el valor de la variable $L_e = w$ se dejará como verdadero, de otro modo falso. La fórmula φ_B es verdadera porque para cada arco e existe sólo una variable $L_e = w$ que es cierta. Las fórmulas φ_C, φ_D y φ_E son verdaderas puesto que son conjunciones de implicancias donde las variables en la premisa están fijas, y las que están en la consecuencia no están fijas. Por lo tanto cada vez que la variable en la premisa sea cierta, a la variable en la consecuencia se le asigna el valor verdadero, el resto de las variables se dejan falsas. Ahora, con esta construcción si el arco e está etiquetado con la palabra w en G la variable $L_e = w$ es verdadera, y para cada prefijo (sufijo) α de w, $L_e = \alpha\Sigma^*$ ($L_e = \Sigma^*\alpha$) es verdadera, y para cada subpalabra α de w, $L_e = \Sigma^*\alpha\Sigma^*$ es verdadera también.

Supongamos que φ_A es falsa, entonces hay una meaningful function f_j, una asignación de palabras para la consulta con una forma de particionarla en $(\bar{w}_1, ..., \bar{w}_m)$, y para cada i tal que $1 \leq i \leq m$ hay un camino $c_i = (e_1, ..., e_{k_i}) \in C_{j,k_i,i}$ tal que $\varphi_{c_i,j,i,(w_l)_l^k}$ es falsa. Si $\varphi_{c_i,j,i,(w_l)_l^k}$ es falsa, en el camino c_i en G es posible evaluar (x_i, w_i, y_i), y esto se tiene para todo i. Lo que significa que para cada i hay un camino c_i en el cual es posible evaluar (x_i, w_i, y_i) con $w_i \in L_i$, luego es posible evaluar la consulta en G. Con esto se tiene una contradicción con el hecho de que $\bar{t} \notin Q(G)$, luego φ_A es verdadera y φ también lo es. Por lo tanto se tiene la equivalencia φ es satisfacible si y sólo si $\bar{t} \notin \Box Q(\pi)$.

Ahora se probará que el número de variables en la fórmula φ es logarítmico en el tamaño de π. Para finalizar la demostración se usará la notación $|\varphi|$ como el tamaño de φ en el número de variables distintas. Sea $K = \max_{\bigcup L_i} |w|$ el largo máximo de palabras en los lenguajes L_i de las conjunciones de la consulta Q, K es constante puesto que los lenguajes L_i en las conjunciones de Q son finitos y Q es constante. En φ_C, φ_D y φ_E se usan sólo variables $L_e = \omega$ que aparecen en φ_A, luego el tamaño de φ_C, φ_D y φ_E como número de variables distintas está acotado por $3|\varphi_A|K$. En φ_B sólo es necesario usar las variables que aparecen en $\varphi_A, \varphi_C, \varphi_D$ y φ_E. Así se tiene que acotar solamente el número de variables en φ_A.

El número de variables en φ_A está acotado por el número de fórmulas $\varphi_{c,j,i,(w_l)_l^k}$ en φ_A por el máximo número de variables en cada $\varphi_{c,j,i,(w_l)_l^k}$. El máximo número de variable en $\varphi_{c,j,i,(w_l)_l^k}$ es $d_{out}(\pi)^K$, y el número de fórmulas es m por el número de *meaningful functions*, luego se tiene $|\varphi_A| \leq Cm \log |\pi| d_{out}(\pi)^K \leq Cm \log |\pi| d^K$, con C una constante. Por lo tanto el número de variables en φ es logarítmico en el tamaño de π.

Como el número de variables en la fórmula es logarítmico en el tamaño de π, y el tamaño total de φ es a lo más polinomial, se puede verificar satisfacibilidad en tiempo polinomial. Luego podemos resolver Certain Answers en tiempo polinomial mediante $\bar{t} \notin \Box Q(\pi) \Leftrightarrow \varphi \in SAT$. □

3.2 CoNP-completitud de los otros casos

En esta sección se muestra que si cualquiera de las condiciones en el Teorema 3.6 no se satisface, el problema de responder las Certain Answers es coNP-completo.

Recordemos las condiciones usadas en el Teorema 3.6, con la misma notación:

($C1$) Si $\bar{t}$ no está en la evaluación de Q sobre $comp(\pi)$, entonces la cantidad de meaningful functions para π, Q y $\bar{t}$, está acotada logarítmicamente por el tamaño de π.

($C2$) $d_{out}(\pi)$ está acotado por una constante dada, dónde $d_{out}(\pi)$ es el grado de salida de π.

($C3$) Los lenguajes L_i de $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ son finitos para cada i.

Teorema 3.9. *Si cualquiera de las condiciones* (C1)*,* (C2) *o* (C3) *no se satisface, el problema de verificar si* $\bar{t} \in \Box Q(\pi)$ *es coNP-completo.*

Demostración. Para probar este teorema se mostrará que si no se tiene cualquiera de las 3 condiciones (C_i) el problema es coNP-duro. Todas las reducciones son del problema de verificar cuando una fórmula proposicional en 3CNF es satisfacible.

1. Caso sin *(C1)*
   Este resultado proviene de [1] con algunas modificaciones. En aras de la completitud se presenta la demostración aquí. Primero, se necesita la siguiente definición:

   Definición 3.10. Decimos que un patrón de grafos π es un *patrón de grafos DAG*, si su estructura de grafo es un grafo DAG (grafo dirigido acíclico).

   Se construirá desde una instancia de 3CNF un patrón de grafos DAG con $d_{out}(\pi)$ acotado y una consulta $Q = (x, L, y)$ con un lenguaje regular L que consiste de una sola palabra, el cual muestra que el problema es coNP-duro si no se tiene un control sobre $mf(Q, \Pi, \bar{t})$.

   Notemos que si se tiene algún arco del patrón de grafos con un lenguaje regular $0|1$, es lo mismo a que ese arco esté etiquetado con una etiqueta variable cuyos posibles valores son 0 y 1. Dado que sólo se utilizaran lenguajes $0|1$ en la demostración, para poder simplificar la notación y expresar las ideas de mejor ya que lo que se busca representar son arcos que pueden optar entre dos valores, a lo largo de la demostración del Teorema 3.9 se utilizará la notación de etiquetas variables.

   Para simplificar las ilustraciones, cuando se presenten patrones con caminos entre dos nodos cuyos arcos son todos completos, esto es, que sus etiquetados son un solo símbolo del alfabeto se ilustrará un arco etiquetado con la palabra que se lee en el camino. Esto ayudará a que las ilustraciones sean más clarificadores, además no genera ningún tipo de dificultad puesto que un camino de arcos completos en un patrón es equivalente a un arco incompleto con una sola palabra, siempre y cuando no existan más caminos cruzándose.

   Sea φ una instancia de 3CNF

$$\varphi = \bigwedge_{i=0}^{n} \psi_i$$

donde $\psi_i = (\phi_1^i \vee \phi_2^i \vee \phi_3^i)$ con $\phi_j^i = x_l$ o $\phi_j^i = \neg x_l$ para algún x_l que es una de las variables $(x_l)_{l=1}^k$ en φ.

Se construye el patrón de grafos π con las siguientes características:

(a) Para cada cláusula $\psi_i = (\phi_1^i \vee \phi_2^i \vee \phi_3^i)$ en φ, se colocan los nodos $i_1, i_2, ..., i_8$. El nodo i_1 es conectado a i_2 por un arco etiquetado con 1, i_2 con i_3 por un arco etiquetado con la variable C_1^i, i_3 con i_4 por un camino etiquetado con 11, i_4 con i_5 por un arco etiquetado con la variable C_2^i, i_5 con i_6 por un camino etiquetado con 111, i_6 con i_7 por un arco etiquetado con la variable C_3^i, i_1 con i_2 por un camino etiquetado con 1111. Obteniendo líneas con la que se muestra en la Figura 3.2.

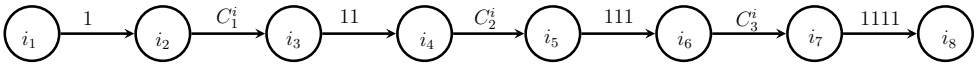

Figura 3.2: Nodos para la cláusula ψ_i

Intuitivamente las variables C_j^i representan los valores de los literales ϕ_j^i.

(b) Sean $p_1, ..., p_k$ las cantidades de ocurrencias totales de las variables $x_1, ..., x_k$ en la fórmula φ, entonces para cada x_l y $j \in \{1, ..., p_l - 1\}$ se construyen los nodos $(l,j)_1, (l,j)_2$ y $(l,j)_3$, conectando $(l,j)_1$ con $(l,j)_2$ por un arco etiquetado con la variable $X_{l,j}$ y el nodo $(l,j)_2$ con $(l,j)_3$ por un camino etiquetado con 1111. Para cada $\phi_n^i = x_l$ en la j-ésima ocurrencia de x_l se conecta i_{2n+1} mediante dos caminos etiquetados por 1101110 y 0110111 al nodo $(l,j)_1$ y mediante otros dos caminos también etiquetados con 1101110 y 0110111 al nodo $(l, j-1)_1$. Si lo nodos $(l, j-1)_1$ o $(l,j)_1$ no existen, no se crean caminos (casos $j = 1$ y $j = p_l$ respectivamente). Para $\phi_n^i = \neg x_l$ se construyen los mismos caminos, pero etiquetados con 110111 y 01101110. Para las cláusulas $\psi_i, \psi_{i'}$ y $\psi_{i''}$ cuando $\phi_1^i = \phi_2^{i'} = x_l$ y $\phi_2^{i''} = \neg x_l$ la estructura resultante para x_l es como en la figura Figure 3.3.

La consulta a usar es la consulta booleana $Q = (x, w, y)$ con $w = 1011011101111$. Ahora se probará $\Box Q(\pi) = F \Leftrightarrow \varphi$ es satisfacible.

Si $\Box Q(\pi)$ es falso, sea $G \in Rep(\pi)$ la GDB en la cual no es posible evaluar (x, w, y). A continuaciones se prueba que el valor de las etiquetas C_h^i y $X_{l,j}$ en G es el mismo cuando $\phi_h^i = x_l$ en φ. Si $C_h^i = 1$ y $X_{l,j} = 0$, hay un camino desde i_{2h+1} hasta $(l,j)_1$ etiquetado por 0110111, tal que desde i_{2h} hasta $(l,j)_3$ evalúa (x, w, y), lo cual no es posible. Si $C_h^i = 0$ y $X_{l,j} = 1$, hay un camino desde i_{2h+1} hasta $(l,j)_1$ etiquetado por 1101110 tal que desde el nodo anterior al i_{2n} hasta $(l,j)_3$ hay un camino etiquetado con 10110111011111 el cual contiene a w, lo cual tampoco es posible.

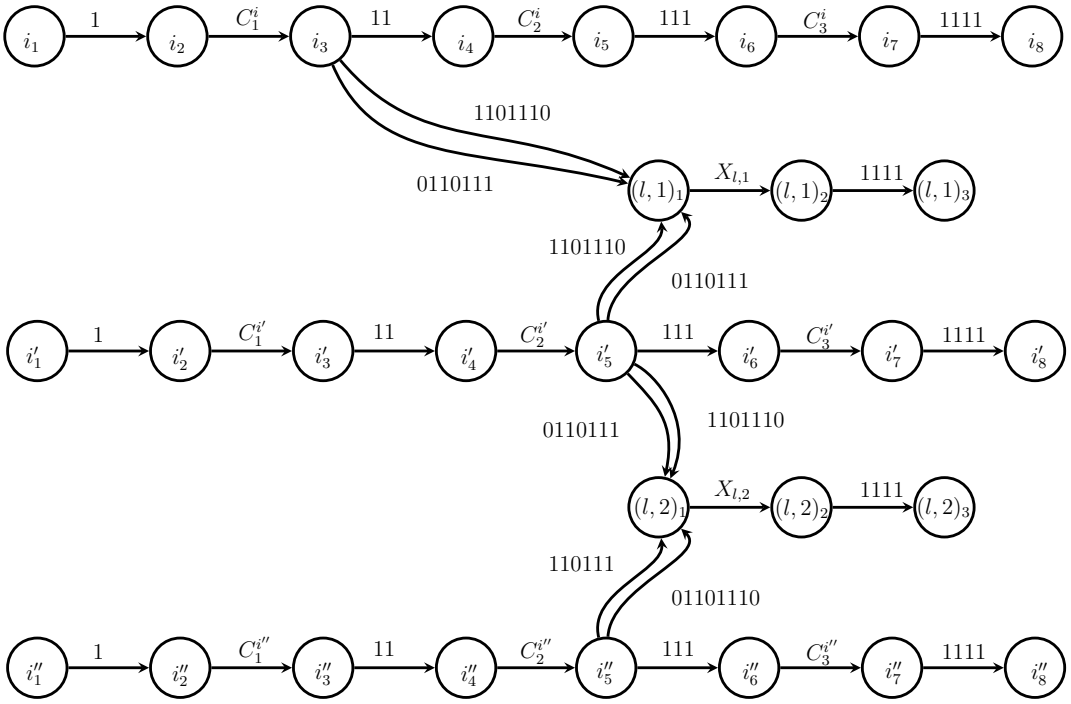

Figura 3.3: Nodos para x_l

A continuación se prueba que C_h^i y $X_{l,j}$ en G tienen valores inversos cuando $\phi_h^i = \neg x_l$ en φ. Si $C_h^i = 1$ y $X_{l,j} = 1$, hay un camino desde i_{2h+1} hasta $(l,j)_1$ etiquetado por 01101110, tal que desde i_{2h} hasta $(l,j)_3$ hay un camino etiquetado por 10110111011111, que contiene a w, lo cual no es posible. Si $C_h^i = 0$ y $X_{l,j} = 0$, hay un camino desde i_{2h+1} hasta $(l,j)_1$ etiquetado por 110111, y desde el nodo anterior al nodo i_{2h} hasta $(l,j)_3$ se puede evaluar la palabra w de Q, lo cual tampoco es posible.

Luego las $X_{l,j}$ tienen el mismo valor para todo j, es el mismo que C_h^i cuando $\phi_h^i = x_l$ y es el valor inverso cuando $\phi_h^i = \neg x_l$. Entonces de G se puede obtener una valuación satisface cada ψ_i, puesto que si no lo hiciera, las etiquetas C_1^i, C_2^i y C_3^i tendrían valor 0 y desde i_1 hasta i_8 se podría evaluar Q. Luego la valuación satisface φ y por lo tanto $\varphi \in SAT$.

Si $\varphi \in SAT$, existe una valuación que satisface φ, sea $G \in Rep(\pi)$ la GDB que tiene en sus etiquetas de arcos C_h^i los valores correspondientes a la valuación que satisface φ y en las etiquetas $X_{l,j}$ los valores de x_l en φ. El resto de la demostración sigue de la demostración del Teorema 3 en [1].

2. Caso sin *(C2)*

Se construirá desde una instancia de 3CNF un patrón de grafos DAG sin $d_{out}(\pi)$ acotado, con $mf(Q, \Pi, \bar{t}) = 1$ y una consulta $Q = (x, L, y)$ con un lenguaje regular L que consiste de una sola palabra, que muestra que el problema es coNP-duro si no se tiene control sobre el $d_{out}(\pi)$.

Sea φ una instancia de 3CNF:

$$\varphi = \bigwedge_{i=0}^{n} \psi_i$$

Donde $\psi_i = (\phi_1^i \vee \phi_2^i \vee \phi_3^i)$ con $\phi_j^i = x_l$ o $\phi_j^i = \neg x_l$ para algún x_l de las variables $(x_l)_{l=1}^k$ en φ.
Se construirá un patrón de grafos π con en el caso anterior, pero con algunas modificaciones:

(a) Por cada cláusula $\psi_i = (\phi_1^i \vee \phi_2^i \vee \phi_3^i)$ en φ, se utilizan los mismos nodos de antes $i_1, i_2, ..., i_8$, con los nuevos nodos $i_{4,2}, i_{6,2}$, y se conectan i_1 con i_2 por un arco etiquetado con 1, i_2 con i_3 por un arco etiquetado con la variable C_1^i, i_3 con $i_{4,2}$ por un arco etiquetado con 1, $i_{4,2}$ con i_4 por un arco etiquetado por 1, i_4 con i_5 por un arco etiquetado con la variable C_2^i, i_5 con $i_{6,2}$ por un camino etiquetado con 11, $i_{6,2}$ con i_6 por un arco etiquetado con 1, i_6 con i_7 por un arco etiquetado con la variable C_3^i, i_7 con i_8 por un camino etiquetado con 1111.

(b) Se crea el nodo 0, y para cada cláusula ψ_i se construyen caminos etiquetados por 000 desde el nodo 0 a los nodos $i_1, i_2, i_{4,2}, i_4, i_{6,2}$ e i_6 como se muestra en la Figura 3.4:

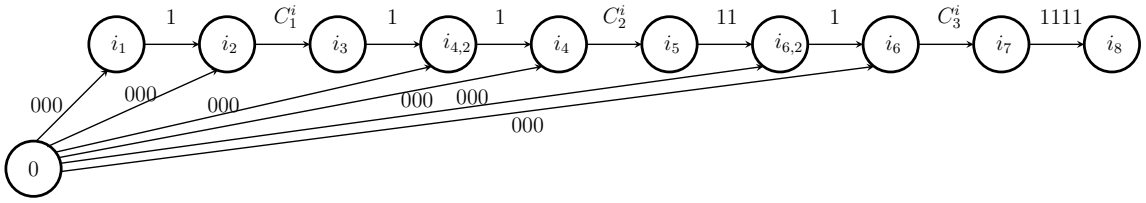

Figura 3.4: Nodo 0

(c) Se crea el nodo f y por cada cláusula C_i se crean caminos etiquetados con 000 desde los nodos i_8 al nodo f, y por cada variable x_l se crean caminos etiquetados con 000 desde los nodos $(l, j)_3$ y los nodos inmediatamente anterior a ellos hasta el nodo f.

La consulta es $Q = (x, w, y)$ con $w = 0001011011101111000$. Entonces se tiene $\Box Q(\pi) = F \Leftrightarrow \varphi$ es satisfacible.

Claramente se obtiene un patrón de grafos DAG con $d_{out}(\pi)$ no acotado. Hay un solo nodo desde el cual se puede evaluar x de la consulta $(x, 0001, y)$, el nodo 0, puesto que incluso si se dejan en 0 todas las etiquetas variables, hay a lo más dos 0s juntos, excepto por los caminos que vienen desde el nodo 0 y los caminos que van hasta el nodo f, pero $d_{out}(f) = 0$. Por la misma razón el único posible nodo de termino es el nodo f, por lo tanto $mf(Q, \Pi, \bar{t}) \leq 1$.

Para completar la demostración se hace una reducción al caso anterior, sea π' el patrón de grafo construido en el caso anterior desde φ y $Q' = (x, w', y)$ con $w' = 1011011101111$, se probará que $\Box Q(\pi)) = F \Leftrightarrow \Box Q'(\pi') = F$.

Si $\Box Q'(\pi') = F$ obviamente se tiene $\Box Q(\pi) = F$ puesto que el patrón de grafo es el mismo, excepto por los nodos 0, f y los caminos 000, los cuales no son un aporte para evaluar una palabra que empieza y termine en 1 y no posee múltiples 0s. Así que si no se puede evaluar (x, w', y) en π', tampoco se puede evaluar (x, w', y) en π y entonces no se puede evaluar $Q = (x, w, y)$ debido a que $w = 000w'000$, luego $certain(Q, \pi) = F$.

Si $\Box Q'(\pi') = V$ entonces es posible evaluar (x, w', y) en π. Notemos que los únicos nodos posibles en que se pueda empezar a evaluar la consulta (candidatos para x) son los nodos $i_1, i_2, i_{4,2}, i_4, i_{6,2}$ e i_6. Como el nodo 0 tiene caminos salientes con 000 precisamente a los nodos $i_1, i_2, i_{4,2}, i_4, i_{6,2}$ e i_6, es posible evaluar la consulta $(x, 000w', y)$. Los únicos nodos en que es posible finalizar la evaluación de la consulta $(x, 000w', y)$ (candidatos a y) son precisamente los nodos que tienen caminos 000 al nodo f, por lo tanto se puede evaluar la consulta $Q = (x, 000w'000, y)$ y entonces $\Box Q(\pi) = V$.

3. Caso sin *(C3)*

Se construirá desde una instancia de 3CNF un patrón de grafos DAG con $d_{out}(\pi)$ acotado, con $mf(Q, \Pi, \bar{t}) = 1$ y una consulta con un lenguaje regular compuesto por infinitas palabras (con la estrella de Kleene), lo que prueba que el problema es coNP-duro si no se tiene un control sobre los lenguajes de la CRPQ que se consulta.

Sea φ una instancia de 3CNF:

$$\varphi = \bigwedge_{i=0}^{n} \psi_i$$

Donde $\psi_i = (\phi_1^i \vee \phi_2^i \vee \phi_3^i)$ con $\phi_j^i = x_l$ o $\phi_j^i = \neg x_l$ para algún x_l de las variables $(x_l)_{l=1}^k$ en φ.
Se construirá un patrón de grafos π con en el caso anterior, pero con algunas modificaciones:

(a) En vez de crear un nodo 0 como en el caso anterior, para cada cláusula ψ_i se crea un nodo adicional $\widehat{i}$ y se crean caminos etiquetados con 000 desde el nodo $\widehat{i}$ a los nodos $i_1, i_2, i_{4,2}, i_4, i_{6,2}$ e i_6.

(b) Se crea un nodo A, y desde él un árbol binario dirigido hacia las hojas, cuyas hojas son los nodos $\widehat{i}$ y cuyos arcos están todos etiquetados con 0, excepto por los dos primeros arcos que salen desde A, los cuales están etiquetados con 1. La Figura 3.5 muestra esta construcción.

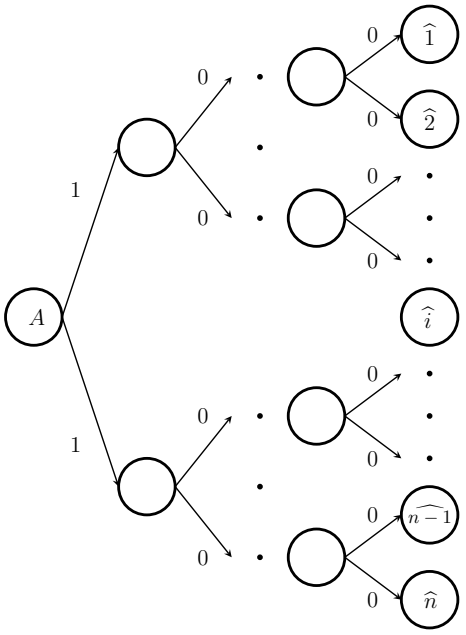

Figura 3.5: Arbol que conecta con los nodos $\widehat{i}$

(c) En vez de crear un nodo f como en el caso anterior, para cada cláusula ψ_i se crea un nodo adicional $\widehat{f_i}$ y se crean caminos etiquetados con 000 desde el nodo i_8 al nodo $\widehat{f_i}$.

(d) Por cada variable x_l se crea un nodo $\widehat{v_l}$ y caminos etiquetados con 000 desde el nodo $(l, j)_3$ y desde el nodo inmediatamente anterior a él, hasta el nodo $\widehat{v_l}$.

(e) Se crea un nodos B, y un árbol binario dirigido hacia la raíz cuyas hojas son los nodos $\widehat{v_l}$ y $\widehat{f_i}$, y cuya raíz es el nodo B. Los arcos se etiquetan todos con 0, excepto por los dos arcos que llegan al nodo B, los cuales están etiquetados con 1.

Sea $Q = (x, L, y)$ la consulta a usar, con $L = 10^*0001011011101111000 0^*1$. Luego se tiene $\Box Q(\pi) = F \Leftrightarrow \varphi$ es satisfacible.

Es claro que el tamaño de los árboles es lineal en el tamaño de φ, luego la GDB incompleta construida tiene tamaño lineal en el tamaño de φ. La GDB construida es DAG, y como se vio en el caso anterior los únicos nodos posibles de partida para comenzar a evaluar 000 son los nodos $\widehat{i}$ (análogos al nodo 0 del caso anterior). En esta nueva construcción también los nodos de los árboles sirven para evaluar 000, pero no sirven para evaluar 00010. Luego el único nodo desde el cual se puede empezar un camino etiquetado con una palabra en 10^*00010 es el nodo A. Del mismo modo, el único nodo posible para terminar la evaluación de $(x, 110000^*1, y)$ (candidato a y) es el nodo B, luego se tiene $mf(Q, \Pi, \bar{t}) \leq 1$.

Para completar la demostración se reducirá a la demostración del caso anterior, así dado el patrón π' que se construye en el caso anterior desde φ y $Q' = (x.w', y)$ con $w' = 0001011011101111000$, se probará que $\Box Q(\pi) = F \Leftrightarrow \Box Q'(\pi') = F$.

Si $\Box Q'(\pi') = F$, entonces no hay un nodo $\widehat{i}$ desde el cual se pueda evaluar $Q' = (x, w', y)$ en π, puesto que si se pudiese evaluar desde un nodo $\widehat{j}$ entonces en π' habría un camino desde el nodo 0 a los nodos correspondientes asociados a la cláusula ψ_j etiquetado por w'.

Las Figuras 3.4 y 3.5 ejemplifican la siguiente parte de la demostración. Suponiendo que desde el nodo A en π se puede evaluar $Q = (x, L, y)$ mediante un camino ρ etiquetado con una palabra $w^* = 10^{n_1}0001011011101111000 0^{n_2}1$ en L, entonces w^* tiene más de un 1, y la única manera de evaluar (x, w^*, y) desde A, con w^* con más de un 1 es a través de un nodo $\widehat{i}$. Porque si no, ρ no alcanza a salir del árbol y su etiquetado sería una palabra en 10^*. Es más, para salir del nodo $\widehat{i}$ la única manera es mediante un camino etiquetado por 000, luego ρ tendría que seguir más allá puesto que al terminar de recorrer 000 aún no pasa por el segundo 1. Como desde $\widehat{i}$ no se puede evaluar (x, w', y) ($\Box Q'(\pi') = F$) y la única manera de salir de $\widehat{i}$ es mediante un camino etiquetado con 000, la única posibilidad para evaluar (x, w^*, y) a través de ρ es que desde $\widehat{i}$ se tenga un camino etiquetado por una palabra en $0^+000w'$, esto es un 0 al menos adicional a los 000 que se recorren en ρ de forma de no evaluar w' que comienza con 0001 desde $\widehat{i}$.

Las Figuras 3.3 y 3.4 ejemplifican lo que viene a continuación. Después de que ρ pasa por $\widehat{i}$, el camino etiquetado por 000, ρ debe pasar a través de un arco etiquetado con 0, por lo tanto los únicos posibles caminos desde $\widehat{i}$ son los que llegan a los nodos i_2, i_4 e i_6. Si ρ pasa por el nodo i_{2k} (y luego al i_{2k+1}), la etiqueta variable C_k^i debe valer 0, luego ρ desde el nodo i_{2k+1} no puede tomar los caminos 1101110 o 0110111 (110111 o 01101110 respectivamente) puesto que en esos casos no estaría evaluando (x, L, y) (w^*

estaría en $10^*11\Sigma^*$). Por lo tanto la única posibilidad es seguir desde el nodo i_{2k+1} al $i_{2(k+1)}$, pero todos los caminos desde nodos i_{2k+1} a $i_{2(k+1)}$ comienzan con 11 de forma que tampoco se estaría evaluando (x, L, y) (w^* estaría en $10^*11\Sigma^*$ también). Luego no hay forma de que ρ esté etiquetado por una palabra en $0^+w'$ desde el nodo $\widehat{i}$, por lo tanto tampoco se puede evaluar (x, L, y) desde A, luego se tiene la contradicción y se concluye $\Box Q(\pi) = F$.

Si $\Box Q'(\pi') = V$ entonces para todo $G \in Rep(\pi)$ hay un nodo $\widehat{i}$ tal que desde $\widehat{i}$ es posible evaluar (x, w', y) en G, luego desde A a $\widehat{i}$ y conectado por un camino que está etiquetado por w', se puede evaluar $(x, 10^*0001011011101111, y)$ mediante un nuevo camino ρ. Es más, el camino ρ termina en un nodo z, y z debe ser i_8, $(l, j)_3$ o el nodo inmediatamente anterior a $(l, j)_3$, luego después de z se puede evaluar $(z, 0000^*1, B)$ finalizando en el nodo B, entonces se puede evaluar (x, L, y) en G. Como se tiene para todo $G \in Rep(\pi)$ se tiene $\Box Q(\pi) = V$.

Este último caso completa la demostración. □

El Teorema 3.9 muestra que nuestra clase tratable es minimal en las condiciones que requiere, puesto que si removemos cualquiera de las condiciones de la clase no se puede resolver el problema de Certain Answers (3.1) en tiempo polinomial.

Corolario 3.11. *Para Q una CRPQ, π una GDB incompleta y $\bar{t}$ una tupla de la aridad de Q, incluso si $\pi \in \mathcal{OD}_{\leq d} \cap \mathcal{MF}^{Q,\bar{t}}_{\leq k}$, el problema de verificar si $\bar{t} \in \Box Q(\pi)$ es coNP-completo.*

Demostración. Directo de la demostración del Teorema 3.9 en el caso sin *(C3)*. □

Capítulo 4

Consultas Eficientes

En esta sección se presenta una forma alternativa de plantear el problema estudiado, convirtiéndolo en un problema de programación lineal entera, bajo la condición de que los lenguajes de consulta sean finitos. Esta representación como problema de programación lineal entera, se basa en identificar, mediante las *meaningful functions*, todos los nodos del patrón en donde sería posible evaluar la consulta. Luego con estos nodos hacer la combinatoria de etiquetas posibles para los arcos que los comunican y establecer como restricciones que los valores de etiquetados no puedan ser los que permiten evaluar la consulta. Así de esta forma obtener una solución si y sólo si hay una asignación de etiquetas en la cual no se puede evaluar la consulta. La demostración de correspondencia entre el problema de Certain Answers y la formulación como problema de programación lineal entera se prueba en el Teorema 4.2.

Luego se hace una variante a esta nueva formulación en donde se eliminan algunas restricciones y el problema se convierte en un problema de optimización. La equivalencia entre el problema de Certain Answers y esta formulación modificada se demuestra en la Proposición 4.4. A ambas formulaciones se les asocian algoritmos heurísticos para resolverlas, los cuales pueden ser detenidos antes de tiempo para obtener una solución candidata en un tiempo deseado. En el capítulo siguiente se presentan condiciones sobre las cuales estos algoritmos son exactos y corren en tiempo polinomial.

Vimos en el capítulo anterior que es posible encontrar una clase tratable para el problema de responder consultas, pero esta clase es restrictiva. Nuestro problema (3.1) se vuelve difícil puesto que en general es coNP-completo y se requieren muchas condiciones para poder resolverlo en tiempo polinomial. Existen técnicas para obtener respuestas, pero pueden no funcionar en límite de tiempo razonable (polinomial). Por lo tanto crearemos algoritmos de aproximación para obtener respuestas con un marco de tiempo razonable.

Los algoritmos probabilísticos usuales proveen soluciones para un problema junto con cotas de error probabilísticas para esas soluciones. No se pueden generar algoritmos de aproximación probabilísticos en este caso puesto que existe una conjetura que establece que

$BPP = P$, donde P es la clase de problemas de decisión que se pueden resolver en forma exacta en tiempo polinomial y BPP es la clase de problemas de decisión que se pueden resolver en tiempo polinomial con probabilidad de error menor a $\frac{1}{3}$. Por lo tanto, como nuestro problema es coNP-completo, encontrar un algoritmo probabilístico con cotas de error hacia un lado o hacia ambos lados sería evidencia en contra de $P \neq NP$.

Lo que se hace en esta tesis es presentar diferentes algoritmos que pueden terminar en tiempo polinomial con soluciones exactas. Sin embargo, existen casos en que los algoritmos pueden ser detenidos antes de tiempo para obtener soluciones aproximadas. En estas situaciones se puede correr varias veces el algoritmo para obtener una solución más confiable, o ejecutar el algoritmo durante un mayor período de tiempo. En el capítulo siguiente se presentan casos para los cuales los algoritmos trabajan en tiempo polinomial entregando una solución exacta.

Como en el capítulo anterior, se continúa trabajando con CRPQs en $\mathcal{F}_{\text{CRPQ}}$ (Conjunctive Regular Patht Queries $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ con L_i finito para todo i). Más adelante se mostrará un algoritmo aproximado para el caso general.

Aquí se presentará un problema de programación lineal entera de factibilidad ($\mathcal{P}$), que se puede construir desde una instancia del problema de Certain Answers. Además, se mostrará que ($\mathcal{P}$) tiene un número polinomial de restricciones y variables en el tamaño de la GDB incompleta. Para probar que existe tal formulación ($\mathcal{P}$) se puede ver simplemente que el problema de Certain Answers es coNP-completo, y el problema de programación lineal entera de factibilidad es NP-completo. Entonces debe existir una reducción en tiempo polinomial entre ambos problemas. Sin embargo, la parte interesante es que la reducción presentada preserva los principales objetos de estudio, las *meaningful functions* como restricciones lineales y los etiquetados posibles de los arcos como variables, entre otros.

Sea $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ una CRPQ con L_i un lenguaje finito $\forall i$, $\pi = (V, E)$ una GDB incompleta, y $\bar{t}$ una tupla de la misma aridad que Q. La reducción para construir ($\mathcal{P}$) es como sigue:

La forma más simple de ver esta reducción es viendo Q como un grafo, donde las variables en $\bar{x}$ e $\bar{y}$ son nodos, y para cada i la RPQ (x_i, L_i, y_i) es un arco entre x_i e y_i etiquetado por L_i.

Recordemos que para palabras α y β, α es *sufijo* de β si β termina en α como por ejemplo $\beta = 011010$ y $\alpha = 010$, α es *prefijo* de β si β comienza con α como por ejemplo $\beta = 011010$ y $\alpha = 011$, además α es *subpalabra* de β si α aparece a partir de cualquier posición en β como por ejemplo $\beta = 011010$ y $\alpha = 1101$. En esta demostración, para α y β palabras, escribiremos $\alpha \sqsubseteq_s \beta$ si α es un sufijo de β, $\alpha \sqsubseteq_p \beta$ si α es un prefijo de β y $\alpha \sqsubseteq \beta$ si α es una subpalabra de β sin utilizar la primera ni la última letra de β

Sea π, Q y $\bar{t}$ como se establecieron en el teorema. Se toman $(w_i)_{i=1}^m$ una asignación de palabras para los lenguajes $(L_i)_{i=1}^m$ de Q, para los cuales existe una cantidad finita de asignaciones diferentes, debido a que los lenguajes L_i son todos finitos.

Se construyen las restricciones $R_{(w_i)_{i=1}^m}$, que se definen de la siguiente forma:

Para cada w_i se toman los enteros $1 \leq k_i \leq |w_i|$, y una descomposición $(w_{i,j})_j$ de w_i en k_i palabras tales que $w_{i,1}w_{i,2}...w_{i,k_i} = w_i$. La cantidad de estas combinaciones es constante porque sólo depende de Q.

Primero se presentará el caso en que las variables existencialmente cuantificadas de Q pueden ser mapeadas solamente a nodos de π. Se puede suponer sin perdida de generalidad que en la consulta Q no hay variables existencialmente cuantificadas que aparecen una sola vez en $\bar{x}$ y una sola vez en $\bar{y}$ (debido a la Proposición 8.4 en el apéndice). Esto simplifica la reducción porque cuando se evalúa la consulta, las únicas variables que pueden ser mapeadas en un arco de π son las existencialmente cuantificadas.

Se cambian los arcos (x_i, L_i, y_i) de Q por caminos en los cuales los arcos están etiquetados por las palabras $w_{i,j}$, obteniendo un nuevo grafo $G = (V_G, E_G)$ con $|E_G| = \sum_i k_i$ y $|V_G| \leq 2|E_G|$, donde $|E_G|$ es acotado por constante (porque depende de Q). Se mapean los nodos de G a los nodos en π, se consideran todas las formas posibles de hacer este mapeo en las cuales las palabras en los arcos de G se pueden mapear en los arcos de π. La cantidad de combinaciones en que podemos escoger los nodos es polinomial $|\pi|^{\sum_i k_i}$. Se deben ignorar las combinaciones que requieren arcos que no existen en π, las que necesitan que dos arcos etiquetados con palabras diferentes deban tener etiquetada la misma palabra, o que un arco deba tener etiquetadas dos palabras diferentes. Si se mapea un arco etiquetado por α a un arco en π etiquetado por L y $\alpha \notin L$ también se ignorará la combinación.

Cuando la identificación de nodos y arcos está finalizada, desde G se puede considerar el grafo $G' = (V_{G'}, E_{G'})$ que se obtiene de la identificación en π, donde sus arcos son arcos en π, etiquetados con las correspondientes $w_{i,j}$ renombradas como w_e. Luego la restricción asociada es:

$$\sum_{e \in E_{G'}} X_{e=w_e} \leq |E_{G'}| - 1.$$

Donde para los arcos que van hacia (vienen de) un nodo de variable extrema además existencialmente cuantificada de Q, en vez de utilizar una variable $X_{e=w_e}$ se utilizará la variable $X_{e=w_e\Sigma^*}$ (respectivamente $X_{e=\Sigma^* w_e}$).

Definición 4.1. Se dice que $\tilde{x}$ es un nodo de variable extrema (o simplemente variable extrema) si $\tilde{x} \in \bar{y}$ y $\tilde{x} \notin \bar{x}$ o si $\tilde{x} \in \bar{x}$ y $\tilde{x} \notin \bar{y}$, en otras palabras $\tilde{x} \in \bar{y} \Delta \bar{x}$ o intuitivamente

viendo Q como un grafo dirigido ((x_i, L_i, y_i) es un arco desde x_i a y_i), $\tilde{x}$ sería un nodo del que solamente salen arcos, o solamente llegan arcos.

Si hay algún arco e tal que sus extremos son ambos nodos de variables extremales existencialmente cuantificadas hay que hacer un cambio un poco más grande. Siguiendo la idea de hacer variables que añaden Σ^* a la palabra que se desea evaluar en el arco, se colocará una variable con Σ^* a ambos lados de la palabra, esto es, en vez de hacer el cambio del párrafo anterior, cambiar la variable $X_{e=w_e}$ por una variable $X_{e=\Sigma^* w_e \Sigma^*}$.

Todas estas restricciones asociadas a una asignación de palabras $(w_i)_{i=1}^m$ serán el grupo de restricciones que forman una $R_{(w_i)_{i=1}^m}$.

Ahora se presenta el caso en que hay nodos que pueden ser mapeados a arcos en π. La idea es ver que sólo algunos nodos pueden ser mapeados a un arco en π. De la Proposición 8.4 en el apéndice, se pueden evitar los nodos existencialmente cuantificados que aparecen sólo una vez en $\bar{x}$ y sólo una vez en $\bar{y}$. Si v es un nodo existencialmente cuantificado que puede ser mapeado a un arco en π, se construye un grafo equivalente que tiene unidos los arcos de salida y los arcos de entrada de v, removiendo el nodo v.

Ahora se muestra cuales son los nodos de variables existencialmente cuantificadas en Q tal que pueden ser mapeados en arcos de π. Para encontrar estos nodos, se buscan variables v tal que para cada i en que la RPQ (x_i, L_i, y_i) cumpla $x_i = v$, se puedan mezclar sus palabras asociadas $w_{i,1}$ (definidas al comienzo de la construccipon de $R_{(w_i)_{i=1}^m}$) para colocarlas en un mismo camino.

Por cada variable existencialmente cuantificada v en la consulta Q, sea $O_v = \{w_{i,1} : x_i = v\}$ el conjunto de palabras iniciales en las RPQs que comienzan con v y sea $I_v = \{w_{i,k_i} : y_i = v\}$ el conjunto de las palabras finales en las RPQs que terminan con v. La idea es tomar las variables tal que cada palabra que empieza en la variable es prefijo de otra, así se toma A el conjunto de variables v tal que satisfacen las primeras dos condiciones, las condiciones impares o las condiciones pares de las siguientes:

1. Existe una palabra w_{O_v} en O_v tal que las demás palabras en O_v son prefijos de w_{O_v}.
2. Existe una palabra w_{I_v} en I_v tal que las demás palabras en I_v son sufijos de w_{I_v}.
3. Existe una única RPQ (x_i, L_i, y_i) en Q tal que $y_i = v$.
4. Existe una única RPQ (x_i, L_i, y_i) en Q tal que $x_i = v$.

Es importante notar que las variables en A son las únicas que pueden ser mapeadas a un nodo dentro de un arco en π, porque es necesario juntar todas las palabras en O_v en una

sola, y todas las palabras en I_v en una sola también. Por cada una de estas variables $v \in A$, se hacen las combinaciones en que se dejan intactas las RPQs de Q como en la demostración anterior, y las combinaciones en que se juntan las RPQs (RPQs de Q vistas como arcos) de la siguiente forma:

Si v satisface la primera condición, sea $(u_j)_{j=1}^k$ el vector de las palabras en O_v, además sea $(b_j)_{j=1}^k$, el vector de los nodos de destino de los arcos en que van las palabras u_j para j con $1 \leq j \leq k$. Se consideran las palabras u_j ordenadas de la más corta a la más larga, y como ellas son todas prefijos de la más larga (u_k), se pueden reescribir las palabras de la siguiente forma:

$$\begin{aligned} u_k &= \alpha_1\alpha_2\alpha_3...\alpha_{k-1}\alpha_k \\ u_{k-1} &= \alpha_1\alpha_2\alpha_3...\alpha_{k-1} \\ &... \\ u_2 &= \alpha_1\alpha_2 \\ u_1 &= \alpha_1 \end{aligned}$$

Se remueven los arcos en O_v, y se agrega un arco que va a b_1 etiquetado por α_1, y así desde b_j se coloca un arco hasta el nodo b_{j+1} etiquetado α_{j+1}. Si v satisface la segunda condición la transformación es análoga. Por ejemplo los nodos en la Figura 4.1 cambian a los nodos en la Figura 4.2.

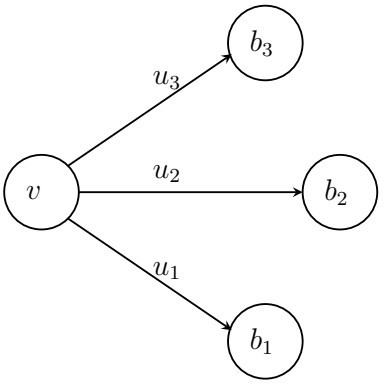

Figura 4.1: Nodos antes de mezclar.

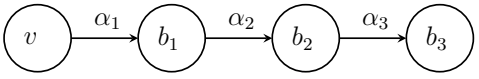

Figura 4.2: Nodos después de mezclar.

En este punto el nodo v tiene a lo más un arco de salida e_O y uno de entrada e_I. Si sólo hay uno de ellos el proceso ha terminado, de lo contrario se remueve el nodo v juntando el arco e_I etiquetado por β con el arco de salida e_O etiquetado por γ, de forma que se reemplazan estos dos arcos por un nuevo arco e que va desde el origen de e_I hasta el destino de e_O etiquetado por $\beta\gamma$.

Haciendo esta transformación para un nodo v, se obtiene una nueva instancia de la consulta, que mapea sus nodos en nodos de π y considera el caso en que v es mapeado a un arco de π.

Se hacen todas las combinaciones con respecto a juntar los arcos de los nodos (variables de Q) en A o no hacerlo, las cuales son finitas puesto que es una cantidad que sólo depende de la consulta y el proceso para hacer estas combinaciones también sólo depende del tamaño de la consulta.

Luego, se obtiene un número polinomial de restricciones por cada $R_{(w_i)_{i=1}^m}$ y la cantidad de restricciones de esta forma sólo depende de la consulta. Si bien estas restricciones son las que imponen la condición de la posibilidad de evaluar la consulta en el patrón, hay que considerar otras restricciones adicionales para no estar considerando casos patológicos y que la solución cumpla con las demás características del problema.

Se necesitan las restricciones que aseguren que cada arco en π posee una sola etiqueta:

$$\forall e \in E \sum_{w \in L_e} X_{e=w} = 1$$

Se necesitan las restricciones que aseguren valores correctos a las variables $X_{e=w_e\Sigma^*}$, $X_{e=\Sigma^* w_e}$ y $X_{e=\Sigma^* w_e \Sigma^*}$:

$$\forall e \in E \quad \forall w \in L_e \quad \forall \alpha \sqsubseteq_p w \quad X_{e=w} \leq X_{e=\alpha\Sigma^*}$$

$$\forall e \in E \quad \forall w \in L_e \quad \forall \alpha \sqsubseteq_s w \quad X_{e=w} \leq X_{e=\Sigma^*\alpha}$$

$$\forall e \in E \quad \forall w \in L_e \quad \forall \alpha \sqsubseteq w \quad X_{e=w} \leq X_{e=\Sigma^*\alpha\Sigma^*}$$

Además se necesitan las restricciones de integralidad:

$$\forall e \in E \quad \forall w \in L_e \quad X_{e=w} \in \{0,1\}$$

$$\forall e \in E \quad \forall w \in L_e \quad X_{e=w\Sigma^*} \in \{0,1\}$$

$$\forall e \in E \quad \forall w \in L_e \quad X_{e=\Sigma^* w} \in \{0,1\}$$

$$\forall e \in E \quad \forall w \in L_e \quad X_{e=\Sigma^* w \Sigma^*} \in \{0,1\}$$

Gracias a la Proposición 3.8 se pueden suponer finitos los lenguajes en los arcos de π. Luego el número de variables es $\sum_{e \in E} |L_e|$, donde L_e es finito, así el número de variables $X_{e=w}$ es lineal en $|E|$, esto es, polinomial en el tamaño de π (lineal). La cantidad de variables $X_{e=\alpha\Sigma^*}$ y $X_{e=\Sigma^*\alpha}$ es polinomial en el tamaño de las palabras en los lenguajes L_i de las RPQs de la consulta Q y la cantidad de palabras en los lenguajes L_e de π, entonces el total de variables es polinomial.

Entonces el problema se puede establecer de la siguiente forma:

($\mathcal{P}$) Factibilidad de:

$$\forall (w_i)_{i=1}^m \in \prod_{i=1}^m L_i \quad R_{(w_i)_{i=1}^m}$$

$$\forall e \in E \qquad \sum_{w \in L_e} X_{e=w} = 1$$

$$\forall e \in E \quad \forall w \in L_e \quad \forall \alpha \sqsubseteq_p w \quad X_{e=w} \leq X_{e=\alpha\Sigma^*}$$
$$\forall e \in E \quad \forall w \in L_e \quad \forall \alpha \sqsubseteq_s w \quad X_{e=w} \leq X_{e=\Sigma^*\alpha}$$
$$\forall e \in E \quad \forall w \in L_e \quad \forall \alpha \sqsubseteq w \quad X_{e=w} \leq X_{e=\Sigma^*\alpha\Sigma^*}$$
$$\forall e \in E \quad \forall w \in L_e \qquad X_{e=w} \in \{0,1\}$$
$$\forall e \in E \quad \forall w \in L_e \qquad X_{e=w\Sigma^*} \in \{0,1\}$$
$$\forall e \in E \quad \forall w \in L_e \qquad X_{e=\Sigma^* w} \in \{0,1\}$$
$$\forall e \in E \quad \forall w \in L_e \qquad X_{e=\Sigma^* w \Sigma^*} \in \{0,1\}$$

Teorema 4.2. *Sea $Q = \bigwedge_{i=1}^m (x_i, L_i, y_i)$ una CRPQ con L_i un lenguaje finito $\forall i$, π una GDB incompleta, y $\bar{t}$ una tupla de la misma aridad que Q. Dada la instancia del problema de programación lineal ($\mathcal{P}$) asociada a Q, π y $\bar{t}$, se tiene:*

$$\bar{t} \notin \Box Q(\pi) \Leftrightarrow (\mathcal{P}) \text{ es factible}$$

Demostración. La demostración consiste en notar que por cada solución de ($\mathcal{P}$) hay una asignación de etiquetas en π que generan una base de datos $G \in Rep(\pi)$ y que las condiciones impuestas en una solución de ($\mathcal{P}$) hacen que no sea posible evaluar Q en G. La demostración de la recíproca se basa en las mismas ideas.

Si $\bar{t} \notin \Box Q(\pi)$, luego existe una GDB $G \in Rep(\pi)$ tal que $\bar{t} \notin Q(G)$. De G se pueden asignar valores a las variables $X_{e=w}$, si w_e es la etiqueta del arco e en G, se toma $X_{e=w_e} = 1$ y $\forall w \neq w_e \; X_{e=w} = 0$. Claramente las restricciones siguientes se satisfacen:

$$\forall w \in \Sigma^* X_{e=w} \in \{0,1\}$$

$$\forall e \in E \sum_{w \in L_e} X_{e=w_e} = 1$$

Las variables $X_{e=w\Sigma^*}$ son 1 si $w \sqsubseteq_p w_e$, las variables $X_{e=\Sigma^* w}$ son 1 si $w \sqsubseteq_s w_e$, las variables $X_{e=\Sigma^* w \Sigma^*}$ son 1 si $w \sqsubseteq w_e$ y 0 en otro caso. Luego las siguientes restricciones se satisfacen:

$$\begin{array}{llllll}
\forall e \in E & \forall w \in L_e & \forall \alpha \sqsubseteq_p w & X_{e=w} \leq & X_{e=\alpha\Sigma^*} \\
\forall e \in E & \forall w \in L_e & \forall \alpha \sqsubseteq_s w & X_{e=w} \leq & X_{e=\Sigma^*\alpha} \\
\forall e \in E & \forall w \in L_e & \forall \alpha \sqsubseteq w & X_{e=w} \leq & X_{e=\Sigma^*\alpha\Sigma^*} \\
\forall w & \in \Sigma^* & X_{e=w\Sigma^*} \in \{0,1\} & & \\
\forall w & \in \Sigma^* & X_{e=\Sigma^* w} \in \{0,1\} & & \\
\forall w & \in \Sigma^* & X_{e=\Sigma^* w\Sigma^*} \in \{0,1\} & &
\end{array}$$

Supongamos que hay una restricción en algún $R_{(w_i)_{i=1}^m}$ que no se satisface, y sea

$$\sum_{e \in E_{G'}} X_{e=w_e} \leq |E_{G'}| - 1$$

la restricción que no se satisface. Entonces se tiene $\forall e \in E_{G'} X_{e=w_e} = 1$ pues las variables son enteras, pero por construcción, las palabras w_e vienen de una asignación de palabras en Q, lo que significa que es posible evaluar Q en G, lo cual es una contradicción con $\bar{t} \notin Q(G)$. Luego la asignación satisface cada restricción, así se tiene una solución para $(\mathcal{P})$ y por lo tanto $(\mathcal{P})$ es factible.

Si $(\mathcal{P})$ tiene una solución, gracias a la restricción:

$$\forall e \in E \sum_{w_e \in L_e} X_{e=w_e} = 1$$

se pueden usar las variables $X_{e=w_e}$ que tienen valor 1 para crear una base de datos $G \in Rep(\pi)$ reemplazando los lenguajes en los arcos de π por las palabras w_e.

Las restricciones:

$$\begin{array}{llllll}
\forall e \in E & \forall w \in L_e & \forall \alpha \sqsubseteq_p w & X_{e=w} \leq & X_{e=\alpha\Sigma^*} \\
\forall e \in E & \forall w \in L_e & \forall \alpha \sqsubseteq_s w & X_{e=w} \leq & X_{e=\Sigma^*\alpha} \\
\forall e \in E & \forall w \in L_e & \forall \alpha \sqsubseteq w & X_{e=w} \leq & X_{e=\Sigma^*\alpha\Sigma^*}
\end{array}$$

aseguran que si $X_{e=w} = 1$, luego $\forall \alpha \sqsubseteq_s w$ $X_{e=\Sigma^*\alpha} = 1$, $\forall \alpha \sqsubseteq_p w$ $X_{e=\alpha\Sigma^*} = 1$ y $\forall \alpha \sqsubseteq w$ $X_{e=\Sigma^*\alpha\Sigma^*} = 1$. Cabe destacar que pudiese ocurrir que la solución obtenida para $(\mathcal{P})$, sin tener un $X_{e=w} = 1$ para algún $\alpha \sqsubseteq_s w$ podría tener $X_{e=\Sigma^*\alpha} = 1$. Si ese es el caso, entonces la misma solución con $X_{e=\Sigma^*\alpha} = 0$ es también una solución porque si $X_{e=\Sigma^*\alpha} = 1$ satisface las restricciones, entonces $X_{e=\Sigma^*\alpha} = 0$ también lo hará.

Supongamos que es posible evaluar la consulta en la base de datos de grafo G, luego existe una asignación $(w_i)_{i=1}^m$ a los arcos de Q la cual hace posible evaluar Q en G. Si hay nodos (variables) en Q que son mapeados en arcos de π, se pueden unir los arcos de salida y los arcos de entrada como se vio en la construcción de $R_{(w_i)_{i=1}^m}$, para obtener una instancia de la consulta que mapea sus nodos, sólo a nodos en π (excepto por los nodos extremos en Q).

Como los nodos en la instancia de Q son mapeados sólo a nodos fijos en π, excepto por los nodos extremos, existe una partición por cada palabra w_i en $(w_{i,j})_{j=1}^{k_i}$, tal que las palabras $w_{i,j}$ son las que están presentes en los arcos de G, excepto los arcos extremos de Q en los cuales la parte final o inicial de la palabra podría no estar en la etiqueta del arco. Sin embargo, estas palabras en los arcos extremos de Q son prefijos o sufijos del correspondiente arco en G.

Sea $G' = (V_{G'}, E_{G'})$ el subgrafo de las instancias de π, G, en el cual la consulta Q coincide, con las palabras de la instancia de la consulta particionadas como $(w_{i,j})_{i,j}$. Se renombran estas palabras como w_e acorde al arco $e \in E_{G'}$ en que están las palabras.

Como las palabras que etiquetan los arcos de la base de datos de grafo G son palabras w tal que $X_{e=w}$ es igual a 1 en la solución, se tiene:

$$\forall e \in E_{G'} \quad X_{e=w_e} = 1.$$

Lo que es una contradicción con las restricciones $R_{(w_i)_{i=1}^m}$, debido a que simultáneamente se tiene:

$$\sum_{e \in E_{G'}} X_{e=w_e} \leq |E_{G'}| - 1 \wedge \sum_{e \in E_{G'}} X_{e=w_e} = |E_{G'}|$$

Luego no es posible evaluar la consulta en G, y finalmente $\bar{t} \notin \Box Q(\pi)$. □

Ahora que se sabe que hay una equivalencia entre $\bar{t} \notin \Box Q(\pi)$ y la factibilidad de $(\mathcal{P})$, se puede tratar de resolver $(\mathcal{P})$ en vez de intentar resolver el problema de responder las consultas directamente. Para dar un algoritmo que resuelva $(\mathcal{P})$ se usará la versión relajada del problema, la cual es el mismo problema removiendo las restricciones de integralidad. Entonces se presenta un algoritmo para determinar cuando $\bar{t} \in \Box Q(\pi)$, resolviendo $(\mathcal{P})$:

Algoritmo 1 Determinar si $\bar{t} \in \Box Q(\pi)$

Require: $(\mathcal{P})$ el problema de programación lineal de Q, π y $\bar{t}$

```
(P̃) ← la versión relajada de (P)
if (P̃) tiene una solución x̃ then
    if x̃ es entera then
        return Falso
    else
```

```
6:          y ← una variable no entera de x̃
7:          (P_1) ← (P) más la restricción y = 0
8:          (P_2) ← (P) más la restricción y = 1
9:          return Algoritmo 1 (P_1) and Algoritmo 1 (P_2)
10:     end if
11: else
12:     return Verdadero
13: end if
```

Teorema 4.3. *Dada* $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ *una CRPQ con* L_i *un lenguaje* $\forall i$, π *una GDB incompleta, y* $\bar{t}$ *una tupla de la misma aridad que* Q. Algoritmo *1 responde* $\bar{t} \in \Box Q(\pi)$.

Demostración. Algoritmo 1 es exacto porque resuelve el problema relajado, en la línea 2 se bifurca entre si el problema relajado $(\tilde{\mathcal{P}})$ es infactible o no. Si $(\tilde{\mathcal{P}})$ es infactible, entonces $(\mathcal{P})$ también es infactible puesto que $(\tilde{\mathcal{P}})$ tiene restricciones más débiles y por lo tanto una región factible más grande que contiene a la región factible de $(\mathcal{P})$. Luego por el Teorema 4.2 se tiene $\bar{t} \in \Box Q(\pi)$ y se responde *Verdadero* en la línea 12. Si $(\tilde{\mathcal{P}})$ es factible y la solución es un vector entero, entonces $(\mathcal{P})$ es factible. Por lo tanto por Teorema 4.2 se tiene $\bar{t} \notin \Box Q(\pi)$, y se responde *Falso* en la línea 4.

Si $(\tilde{\mathcal{P}})$ es factible y la solución no es un vector entero, se bifurca en cada posible combinación de valores. Si uno de los problemas es factible entero (ellos responden *Falso*), y se responde *Falso* en la línea 9 porque $(\mathcal{P})$ es factible. Si ambos problemas son infactible (ellos responden *Verdadero*), se responde *Verdadero* en la línea 9 puesto que $(\mathcal{P})$ es infactible. □

En la línea 6, si se escoge la variable más grande (o algún otro criterio), se salta la línea 7 y en la 8 sólo se retorna *Algoritmo* 1 aplicado a $(\mathcal{P}_2)$ se obtiene una heurística. En *Algoritmo* 1 el número de bifurcaciones podría llegar a ser exponencial en el peor caso. Sin embargo, como hay un número polinomial de variables y cada iteración toma tiempo polinomial, se obtiene una solución en tiempo polinomial usando la versión heurística.

Es importante destacar que en la versión heurística se puede cambiar el criterio para escoger la variable a fijar y así poder explorar diferentes acercamientos.

En el siguiente ejemplo presentamos una base de datos de tamaño $\mathcal{O}(n)$, en la cual sin ayuda del *Algoritmo* 1 intentando por fuerza bruta tardaríamos tiempo exponencial en n en responder Certain Answers.

Ejemplo 4.1. Sea $Q = (x, \{011\}, y)$ una CRPQ y $\pi = (V, E)$ una GDB incompleta, donde $V = \{a, b_1, b_2, ..., b_n, c_1, c_2, ..., c_n, d\}$ y E está compuesto por los arcos $e_1, e_2, ..., e_n$, $f_1, f_2, ..., f_n$ y $g_1, g_2, ..., g_n$, dónde los arcos e_i van desde el nodo a a los nodos b_i etiquetados por 0, los argos f_i van desde los nodos b_i a los nodos c_i etiquetados por el lenguaje regular L, y los

arcos g_i van desde los nodos c_i al nodo d etiquetados por 1 como se muestra a continuación en la Figura 4.3.

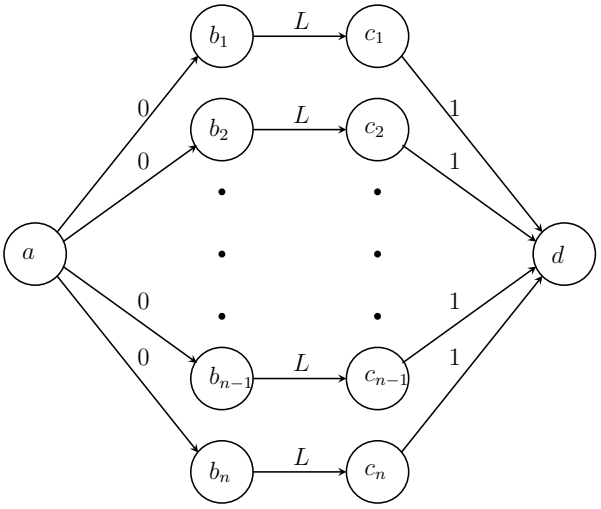

Figura 4.3: GDB Incompleta π.

El tamaño del problema es $\mathcal{O}(n)$ porque $|V| \in \mathcal{O}(n)$. Tomando $L = \{11, 01\}$, el problema pareciera no ser tan difícil. Sin embargo, tiene muchas *meaningful functions*, debido a que se puede evaluar la consulta en muchos caminos (a, b_i, c_i) y (b_i, c_i, d). Además, si π fuese sólo una parte de la base de datos de grafo incompleta a estudiar, podría ser muy complejo notar que si L toma el valor 11 se puede evaluar la consulta en los caminos (a, b_i, c_i), y si L toma el valor 01 también es posible evaluar la consulta, esta vez en los caminos (b_i, c_i, d). A pesar de esto, con *Algoritmo* 1 se puede resolver el problema en tiempo polinomial con respecto a n sin tener que verificar las condiciones previas en π. Puesto que se obtienen las siguientes ecuaciones, entre otras:

$$\begin{aligned}
\forall i \quad & X_{e_i=0} + X_{f_i=11} \leq 1 \\
\forall i \quad & X_{f_i=01} + X_{g_i=1} \leq 1 \\
\forall i \quad & X_{e_i=o} = 1 \\
\forall i \quad & X_{g_i=1} = 1 \\
\forall i \quad & X_{f_i=11} + X_{f_i=01} = 1
\end{aligned}$$

Usando las primeras dos ecuaciones se tiene:

$$\forall i \quad X_{f_i=11} \leq 0$$

$$\forall i \quad X_{f_i=01} \leq 0$$

$\Rightarrow$

$$\forall i \quad X_{f_i=11} = 0 \wedge X_{f_i=01} = 0$$

Lo cual es una contradicción con $X_{f_i=11} + X_{f_i=01} = 1$. Lo que significa que le problema relajado es infactible, por lo que se puede resolver en tiempo polinomial (con *Algoritmo* 1) y se responde $\bar{t} \in \Box Q(\pi)$.

Ejemplo 4.2. Se considera la misma consulta Q y la GDB incompleta π del Ejemplo 4.1, pero el lenguaje $L = \{11, 01, 00\}$ en π. El problema sigue en $\mathcal{O}(n)$, porque el tamaño de π es el mismo. esta vez se tienen ecuaciones diferentes:

$$\forall i \quad X_{f_i=11} + X_{f_i=01} + X_{f_i=00} = 1$$

Luego, como se vio en el Ejemplo 4.1 se tiene $\forall i \; X_{f_i=11} = 0 \wedge X_{f_i=01} = 0$ y ahora $X_{f_i=00} = 1$, esta es la única solución en las variables $X_{e=w}$, y es claro que de ellas se puede obtener una solución entera en las demás variables. Así, se tiene una solución entera, y hay una sola solución (excepto por las variables $X_{e=\Sigma^* w}, X_{e=w\Sigma^*}$ y $X_{e=\Sigma^* w\Sigma^*}$), luego el problema relajado obtiene (en tiempo polinomial) una solución entera, porque hay una sola, por lo tanto se tiene $\bar{t} \notin \Box Q(\pi)$. Lo cual es el mismo resultado que se obtiene con *Algoritmo* 1, pero en tiempo polinomial a pesar de que el algoritmo puede correr en tiempo exponencial (debido a que la solución entera se obtiene de inmediato). En el siguiente capítulo se presentan algunos casos en los que *Algoritmo* 1 corre en tiempo polinomial.

4.1 Algoritmo de Optimización

Como se vio en la sección anterior, se puede formular el problema de Certain Answers como un problema de factibilidad de programación lineal entera, y lo que se hace es resolver la versión relajada del problema para obtener un algoritmo exacto y una heurística. En programación lineal la parte más difícil es verificar factibilidad de una solución. Luego una mejora para nuestro algoritmo sería simplificar las restricciones, incluso si esto implica un costo al tener una función objetivo.

A continuación se muestra que se puede establecer el problema de Certain Answers como un problema de optimización de programación lineal entera ($\mathcal{P}'$) con menos restricciones que ($\mathcal{P}$).

Dada $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ una CRPQ con L_i un lenguaje finito $\forall i$ ($Q \in \mathcal{F}$), $\pi = (V, E)$ una GDB incompleta y $\bar{t}$ una tupla de nodos de la misma aridad que Q se toma la formulación del problema de Certain Answers ($\mathcal{P}$) y se hacen algunas modificaciones.

La idea es modificar las restricciones:

$$\forall e \in E \sum_{w \in L_e} X_{e=w} = 1.$$

En la práctica, para resolver el problema es necesario escribir las restricciones del problema como $Ax \leq b$, donde A es una matriz con tantas columnas como variables en el problema, tantas filas como restricciones y x es el vector de variables del problema. Luego es necesario reescribir las restricciones anteriores como las siguientes restricciones combinadas:

$$\begin{aligned} (1) \quad \forall e \in E \quad & \sum_{w \in L_e} X_{e=w} \leq \quad 1 \\ (2) \quad \forall e \in E \quad & -\sum_{w \in L_e} X_{e=w} \leq -1 \end{aligned}$$

Precisamente, estas son las restricciones que es necesario trabajar en $(\mathcal{P})$. Para formular $(\mathcal{P}')$ se usan las mismas restricciones que en $(\mathcal{P})$, pero sin incluir las restricciones (1) e incluyendo la siguiente función objetivo:

$$f(x) := \sum_{e \in E} \sum_{w \in L_e} X_{e=w}$$

Proposición 4.4. *Dada $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ una CRPQ con L_i un lenguaje finito $\forall i$ $(Q \in \mathcal{F})$, $\pi = (V, E)$ una GDB incompleta y $\bar{t}$ una tupla de nodos de la misma aridad que Q, el problema de Certain Answers (3.1), determinar cuando $\bar{t} \notin \Box Q(\pi)$, es equivalente a resolver $(\mathcal{P}')$, esto es:*

$$\bar{t} \notin \Box Q(\pi) \Leftrightarrow \text{mín}(\mathcal{P}') = |E|$$

Demostración. Claramente se tiene $f(x) = |E| \Leftrightarrow (1)$ bajo la condición (2). Por lo tanto se tiene que, si al resolver $(\mathcal{P}')$ se obtiene un valor $|E|$ significa que $(\mathcal{P})$ tiene solución, si se obtiene un valor estrictamente mayor a $|E|$, significa que $(\mathcal{P})$ es infactible, puesto que si tuviese solución, entonces la misma solución en $(\mathcal{P}')$ entregaría un valor $|E|$ a la función objetivo. □

Ahora se tiene otra formulación de programación lineal entera para el problema de Certain Answers, la cual es análoga a la formulación anterior. Entonces es razonable pensar que debiese existir un algoritmo análogo al algoritmo anterior.

Algoritmo 2 Determinar si $\bar{t} \in \Box Q(\pi)$

Require: $(\mathcal{P}')$ el problema de optimización de programación lineal entera de Q, π y $\bar{t}$

```
(P̃') ← la versión relajada de (P')
if (P̃') tiene argmin x̃ and f(x̃) = |E| then
    if x̃ es entero then
        return Falso
    else
        y ← una variable no entera de x̃
```

7: $(\mathcal{P}'_1) \leftarrow (\mathcal{P}')$ más la restricción $y = 0$
8: $(\mathcal{P}'_2) \leftarrow (\mathcal{P}')$ más la restricción $y = 1$
9: **return Algoritmo 2** $(\mathcal{P}'_1)$ **and Algoritmo 2** $(\mathcal{P}'_2)$
10: **end if**
11: **else**
12: **return** Verdadero
13: **end if**

Teorema 4.5. *Dada* $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ *una CRPQ con* L_i *un lenguaje finito* $\forall i$, π *una GDB incompleta, y* $\bar{t}$ *una tupla de nodos de la misma aridad que* Q. Algoritmo 2 *determina si* $\bar{t} \in \Box Q(\pi)$.

Demostración. La demostración es exactamente la misma a la del Teorema 4.3 utilizando la Proposición 4.4, excepto esta vez, para responder $\bar{t} \in \Box Q(\pi)$ en la línea 12 se necesita $(\mathcal{P}')$ infactible o $f(\tilde{x}) > |E|$. □

Capítulo 5

Garantías

En este capítulo se presentan algunos resultados que muestran en cuales casos los algoritmos anteriores trabajan en tiempo polinomial. Todas las garantías que se presentan en este capítulo son para el problema de Certain Answers donde la consulta es una CRPQ en $\mathcal{F}_{\text{CRPQ}}$. Es necesario recordar que la condición $Q \in \mathcal{F}_{\text{CRPQ}}$ significa que la consulta Q es una CRPQ de la forma $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ donde cada L_i es un lenguaje finito. En el siguiente capítulo se trabaja el caso general en que $Q \notin \mathcal{F}_{\text{CRPQ}}$.

En la primera sección se muestra que bajo las condiciones usadas para probar que se puede computar Certain Answers en tiempo polinomial en el Capítulo 3 (Teorema 3.6) también se puede resolver el problema de forma exacta en tiempo polinomial con los algoritmos del capítulo anterior. Esto se prueba en el Teorema 5.1.

En la segunda sección se muestra que si al escribir la formulación del problema como problema de programación lineal se tiene cierta estructura en la matriz de restricciones (total unimodularidad), los algoritmos de la sección anterior también corren en tiempo polinomial. Los principales resultados son los Teoremas 5.3 y 5.5.

En la tercera sección se entrega una técnica para detectar cuando hay una pequeña porción de la base de datos en la cual siempre se puede evaluar la consulta, luego cumplida la condición, el problema puede ser resuelto en forma exacta y en tiempo polinomial por los algoritmos del capítulo anterior. Esto se prueba en el Teorema 5.7.

A continuación se verá que los algoritmos de aproximación presentados en el Capítulo anterior son robustos, porque ellos pueden computar en tiempo polinomial las Certain Answers del caso anterior del Teorema 3.6.

5.1 Garantías en la Forma del Patrón

En el siguiente Teorema se muestra que los algoritmos de aproximación funcionan en tiempo polinomial y son correctos, bajo las condiciones del Teorema 3.6. Esto indica que nuestros algoritmos de aproximación son robustos en el sentido de que pueden resolver el caso tratable anterior como algoritmos exactos.

Teorema 5.1. *Dados valores fijos $d, k \geq 0$, los problemas $(\mathcal{P})$ y $(\mathcal{P}')$ se pueden resolver en tiempo polinomial con* Algoritmo *1 y* Algoritmo *2 cuando la entrada se restringe a CRPQs $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ donde los lenguajes L_i tienen un número finito de palabras $\forall i$, y pares $(\pi, \bar{t})$ tal que π en $\mathcal{OD}_{\leq d} \cap \mathcal{MF}_{\leq k}^{Q,\bar{t}}$ o se tiene $\bar{t} \in Q(comp(\pi))$.*

Demostración. Si $\bar{t} \in Q(comp(\pi))$ hay restricciones de la forma $R_{(w_i)_{i=1}^m}$ que usan solamente arcos de $(comp(\pi))$, luego cada lenguaje asociado a estos arcos en π contiene una única palabra. Por lo tanto sólo existe una única variable $X_{e=w}$ asociada a cada arco, la cual sólo puede tomar el valor 1. Como la consulta se puede evaluar usando sólo estos arcos, pues $\bar{t} \in Q(comp(\pi))$, se tiene que las restricciones mencionadas al principio no se satisfacen, pues todas sus variables valen 1. Finalmente, al resolver el problema relajado se obtiene inmediatamente la infactibilidad.

Lo importante es ver que el número de variables de la forma $X_{e=w}$ en el problema es logarítmico en el tamaño de la base de datos de grafo incompleta π. El número de variables $X_{e=\Sigma^*\alpha}$ y $X_{e=\alpha\Sigma^*}$ es polinomial en el tamaño de π, pero ellas no son relevantes, puesto que son determinadas automáticamente por el valor de las variables $X_{e=w}$. Para acotar la cantidad de variables se buscará acotar la cantidad de restricciones por el número de *meaningful functions*, y luego acotar por constantes la cantidad de variables en cada restricción.

Notemos que cada restricción en un grupo de restricciones $R_{(w_i)_{i=1}^m}$ está asociada a alguna *meaningful function*, y por cada *meaningful function* hay un número acotado de restricciones, el cual depende del tamaño del la consulta y de $d_{out}(\pi)$.

Por construcción de las restricciones en $R_{(w_i)_{i=1}^m}$, cada restricción viene de una instancia de la consulta mapeando los nodos existencialmente cuantificados (variables existencialmente cuantificadas) de Q a nodos y arcos de π. Además la existencia de la restricción implica que es posible evaluar la consulta en los arcos de π con los etiquetados indicados por las variables de la restricción. Por lo tanto debe existir una *meaningful function* que mapea a los nodos o arcos asociados a la restricción.

Para cada *meaningful function* se pueden asociar las restricciones que se generar mapeando los nodos de la consulta a los nodos y arcos del patrón que mapea la *meaningful function*. Dada una *meaningful function* f hay una cantidad constante de restricciones asociadas a f, debido a que el número de restricciones es a lo más el número de posibles caminos etiquetados con palabras en Q. Esta cantidad de caminos depende de la cantidad y del tamaño de

las palabras en Q y del $d_{out}(\pi)$, luego es acotado por una constante.

Entonces el número de restricciones en todas las $R_{(w_i)_{i=1}^m}$ es logarítmico en el tamaño de π, porque hay una cantidad logarítmica de *meaningful functions* y por cada una de ellas hay una cantidad acotada por constante de restricciones. Además, el número de variables en cada restricción es acotado por una constante, puesto que la cantidad de variables diferentes en una restricción depende del tamaño de loas palabras en la consulta y de la cantidad de RPQs en la consulta. Luego, se tiene que las restricciones de la forma $R_{(w_i)_{i=1}^m}$ usan una cantidad logarítmica de variables diferentes.

Como el número de variables diferentes es logarítmico, la cantidad de arcos en π involucrados también lo es, luego se puede considerar una cantidad logarítmica de restricciones de la forma:

$$\forall e \in E, \sum_{w \in L_e} X_{e=w_e} = 1$$

de manera de que la cantidad de variables en todas las restricciones sean logarítmicas. Así el problema de Certain Answers se puede resolver en tiempo polinomial usando *Algoritmo 1* y *Algoritmo 2*. □

Es importante notar que para asegurar una ejecución en tiempo polinomial de los algoritmos, no es necesario encontrar las *meaningful functions*, sólo es necesario saber que hay una cantidad logarítmica de ellas.

5.2 Garantías de Total Unimodularidad

Para continuar son necesarias algunas definiciones de matrices:

Definición 5.2. Sea M una matriz cuadrada, se dice que M es una matriz *unimodular* si su determinante es 1 ó -1. Además si M es una matriz cualquiera, no necesariamente cuadrada, se dice que M es una matriz *totalmente unimodular* si para cualquier submatriz S de M invertible, S es unimodular.

En esta sección se hará mención de la matriz de restricciones refiriéndose siempre a la matriz usual de programación lineal, que se forma con los coeficientes de las restricciones de un problema de programación lineal.

En esta sección se mostrará que hay algunas restricciones que si ellas forman una matriz totalmente unimodular, el conjunto completo de restricciones forma una matriz totalmente unimodular, y luego cualquier solución del problema (vértice del poliedro generado por las restricciones) es entera. Luego la versión relajada de $(\mathcal{P}')$ obtiene una solución en tiempo

polinomial.

La mayoría de las restricciones en $(\mathcal{P}')$ forman una matriz identidad, así que ellas no son relevantes para verificar la total unimodularidad del sistema. Las otras restricciones son:

$$\begin{aligned}
&\forall (w_i)_{i=1}^m \in \prod_{i=1}^m L_i && R_{(w_i)_{i=1}^m} \\
&\forall e \in E && -\sum_{w \in L_e} X_{e=w} \leq -1 \\
\forall e \in E \quad &\forall w \in L_e \quad \forall \alpha \sqsubseteq_p w && X_{e=w} \quad \leq X_{e=\alpha\Sigma^*} \\
\forall e \in E \quad &\forall w \in L_e \quad \forall \alpha \sqsubseteq_s w && X_{e=w} \quad \leq X_{e=\Sigma^*\alpha} \\
\forall e \in E \quad &\forall w \in L_e \quad \forall \alpha \sqsubseteq w && X_{e=w} \quad \leq X_{e=\Sigma^*\alpha\Sigma^*}
\end{aligned} \tag{5.1}$$

Entonces sólo se necesita que estas restricciones formen una matriz totalmente unimodular.

Teorema 5.3. *Si en las restricciones de $(\mathcal{P}')$, las restricciones en (5.1) forman una matriz totalmente unimodular, determinar si $\bar{t} \in \Box Q(\pi)$ puede ser resuelto en tiempo polinomial por* Algoritmo 2.

Demostración. Por el Teorema 4.4 sólo es necesario resolver $(\mathcal{P}')$ para resolver el problema de Certain Answers. Para resolver la versión relajada del problema hay que cambiar las restricciones:

$$\begin{aligned}
\forall e \in E \quad &\forall w \in L_e \quad X_{e=w} \in \{0,1\} \\
\forall e \in E \quad &\forall w \in L_e \quad X_{e=w\Sigma^*} \in \{0,1\} \\
\forall e \in E \quad &\forall w \in L_e \quad X_{e=\Sigma^*w} \in \{0,1\} \\
\forall e \in E \quad &\forall w \in L_e \quad X_{e=\Sigma^*w\Sigma^*} \in \{0,1\}
\end{aligned}$$

por:

$$\begin{aligned}
\forall e \in E \quad &\forall w \in L_e \quad X_{e=w} \leq 1 \\
\forall e \in E \quad &\forall w \in L_e \quad X_{e=w\Sigma^*} \leq 1 \\
\forall e \in E \quad &\forall w \in L_e \quad X_{e=\Sigma^*w} \leq 1 \\
\forall e \in E \quad &\forall w \in L_e \quad X_{e=\Sigma^*w\Sigma^*} \leq 1 \\
\forall e \in E \quad &\forall w \in L_e \quad -X_{e=w} \leq 0 \\
\forall e \in E \quad &\forall w \in L_e \quad -X_{e=w\Sigma^*} \leq 0 \\
\forall e \in E \quad &\forall w \in L_e \quad -X_{e=\Sigma^*w} \leq 0 \\
\forall e \in E \quad &\forall w \in L_e \quad -X_{e=\Sigma^*w\Sigma^*} \leq 0
\end{aligned} \tag{5.2}$$

Se sabe que si las restricciones de un problema de programación lineal son $Ax \leq b$, donde b es un vector entero y A es una matriz totalmente unimodular, se tiene que cada vértice del

poliedro es entero, por lo tanto el óptimo es una solución entera.

Claramente en $(\mathcal{P}')$ el vector b es entero, y por tanto sólo se necesita verificar que la matriz de restricciones sea totalmente unimodular. Se sabe que para una matriz $A = [A'I]$, donde A' es otra matriz e I es la matriz identidad se tiene que A es totalmente unimodular si y sólo si A' es totalmente unimodular. Lo mismo ocurre con $A = [A' - I]$.

Como las restricciones en (5.2) generan una matriz identidad y una matriz identidad negativa, se pueden ignorar al verificar la total unimodularidad de la matriz de restricciones en $(\mathcal{P}')$. Por lo tanto si se tiene la total unimodularidad de la matriz generada por la matriz de restricciones de la hipótesis, se tendrá que cada vértice del poliedro es entero, luego el óptimo del problema relajado será entero, el cual se puede obtener en tiempo polinomial. Así *Algoritmo 2* entrega la solución correcta en el primer paso al resolver el problema relajado. □

Algunas de las restricciones en $(\mathcal{P}')$, tales como:

$$\begin{aligned} \forall e \in E \quad \forall w \in L_e \quad \forall \alpha \sqsubseteq_p w \quad X_{e=w} \leq X_{e=\alpha\Sigma^*} \\ \forall e \in E \quad \forall w \in L_e \quad \forall \alpha \sqsubseteq_s w \quad X_{e=w} \leq X_{e=\Sigma^*\alpha} \\ \forall e \in E \quad \forall w \in L_e \quad \forall \alpha \sqsubseteq w \quad X_{e=w} \leq X_{e=\Sigma^*\alpha\Sigma^*} \end{aligned} \tag{5.3}$$

no son tan difíciles de evitar, puesto que si los valores de las variables $X_{e=\alpha\Sigma^*}$, $X_{e=\Sigma^*\alpha}$ y $X_{e=\Sigma^*\alpha\Sigma^*}$ son siempre 1, las restricciones se satisfacen. Existen casos en que esas restricciones son simplemente innecesarias, por ejemplo el caso en que los lenguajes en los arcos de la GDB incompleta π sólo tienen palabras de largo 1. De manera formal:

Definición 5.4. Diremos que las restricciones en (5.3) *son irrelevantes*, si en las restricciones de la forma $R_{(w_i)_{i=1}^m}$ las únicas variables de la forma $X_{e=\alpha\Sigma^*}$, $X_{e=\Sigma^*\alpha}$ y $X_{e=\Sigma^*\alpha\Sigma^*}$ que aparecen obligatoriamente tienen que tomar el valor 1.

Teorema 5.5. *Si al establecer $(\mathcal{P}')$, las restricciones en (5.3) son irrelevantes, y además si en las restricciones de la forma $R_{(w_i)_{i=1}^m}$ se tiene que las variables aparecen una sola vez entre todas las restricciones $R_{(w_i)_{i=1}^m}$, determinar si $\bar{t} \in \Box Q(\pi)$ puede ser resuelto en tiempo polinomial con* Algoritmo *2.*

Demostración. Para una matriz A con todas sus componentes en $\{-1, 0, 1\}$ verificar total unimodularidad es equivalente a, dado cualquier subconjunto $M = \{r_1, ..., r_k\}$ de filas, existe una asignación de signos $S : M \to \{1, -1\}$ tal que la fila:

$$\sum_{i=1}^{k} S(r_i) r_i$$

tiene todas sus columnas en $\{-1, 0, 1\}$. Como las variables aparecen a lo más una vez en las restricciones $R_{(w_i)_{i=1}^m}$, y en las restricciones $\forall e \in E - \sum_{w \in L_e} X_{e=w} \leq -1$ las variables

aparecen sólo una vez y con signo negativo, se puede usar cualquier subconjunto M y $S = 1$ constante. Así la matriz de restricciones:

$$\forall (w_i)_{i=1}^m \in \prod_{i=1}^m L_i \quad R_{(w_i)_{i=1}^m}$$

$$\forall e \in E - \sum_{w \in L_e} X_{e=w} \leq -1$$

es totalmente unimodular, y por Teorema 5.3 se tiene que la matriz completa de restricciones es totalmente unimodular. Concluimos que *Algoritmo 2* funciona en tiempo polinomial. □

Hay que notar que si no se utilizan las restricciones en (5.3) y esas restricciones no son *irrelevantes*, se podría obtener una respuesta equivocada.

Observación Es importante notar que se pueden ignorar algunas de las restricciones en (5.3). Porque si las variables $X_{e=\Sigma^*\alpha}$ o $X_{e=\alpha\Sigma^*}$ no aparecen en las restricciones $R_{(w_i)_{i=1}^m}$, se pueden simplemente ignorar sus restricciones asociadas. Esto se debe a que después de obtener la solución entera se pueden obtener valores para las variables $X_{e=\Sigma^*\alpha}$ desde las variables $X_{e=w}$ o simplemente darles el valor 1 (siempre se puede dar el valor $X_{e=\Sigma^*\alpha} = 1$ para satisfacer las restricciones (5.3)).

Observación Se puede crear un algoritmo de heurística, removiendo las restricciones en (5.3) y quitando algunas de las restricciones en

$$\forall (w_i)_{i=1}^m \in \prod_{i=1}^m L_i \quad R_{(w_i)_{i=1}^m}$$

hasta que la matriz de restricciones sea totalmente unimodular, luego resolviendo el problema relajado se obtendrán sólo soluciones enteras, si hay muchas se pueden revisar algunas solamente. Luego restaurar las restricciones eliminadas y probar si la solución las satisface. Si cualquier solución las satisface se puede responder $\bar{t} \notin \Box Q(\pi)$ sin riesgo de error, de lo contrario se puede responder $\bar{t} \in \Box Q(\pi)$ con el riesgo de cometer una equivocación si no se utilizaron todas las soluciones obtenidas anteriormente. Por otro lado si el problema con menos restricciones no tiene solución, es decir, es infactible, con mayor razón lo será con todas las restricciones y por tanto se puede responder $\bar{t} \in \Box Q(\pi)$ sin riesgo de error.

El siguiente ejemplo muestra un problema de Certain Answers, que en principio no podemos saber si se puede computar en tiempo polinomial, pero gracias al Teorema 5.5 es posible asegurar que se pueden computar las Certain Answers en tiempo polinomial con el *Algoritmo 2*.

Ejemplo 5.1. Se considera la misma consulta Q y la misma base de datos de grafo incompleta π del Ejemplo 4.1, pero con el lenguaje $L = \{11, 01, 00, 000\}$ en π. Se tienen las mismas ecuaciones, excepto por los arcos f_i:

$$\forall i \quad X_{f_i=11} + X_{f_i=01} + X_{f_i=00} + X_{f_i=000} = 1.$$

Entonces se tiene $X_{f_i=00} + X_{f_i=000} = 1$, esta vez se tienen soluciones no enteras, para cualquier $0 \leq \lambda_i \leq 1$, se tiene una solución $X_{f_i=11} = 0, X_{f_i=01} = 0, X_{f_i=00} = \lambda_i$ y $X_{f_i=000} = 1 - \lambda_i$, luego el problema relajado tiene soluciones, no son todas enteras.

Como no hay restricciones de la forma (5.3) y las variables aparecen sólo una vez en las restricciones:

$$\forall (w_i)_{i=1}^m \in \prod_{i=1}^m L_i \quad R_{(w_i)_{i=1}^m}$$

entonces se puede aplicar el Teorema 5.5, luego se puede resolver el problema en tiempo polinomial debido a que la matriz de restricciones es totalmente unimodular. Como se sabe que hay soluciones (las recién encontradas), y que *Algoritmo 2* en este caso encuentra soluciones enteras o se topa con infactibilidad, se tiene que hay soluciones enteras y *Algoritmo 2* correrá en tiempo polinomial indicando $\bar{t} \notin \Box Q(\pi)$.

Esto se debe a que removiendo las restricciones de $X_{e=\alpha\Sigma^*}$ se tiene un poliedro con vértices enteros, el espacio de soluciones tiene las variables fijas salvo por $X_{f_i=00} = \lambda_i$ y $X_{f_i=000} = 1 - \lambda_i$ lo que genera un poliedro con vértices enteros ($\lambda_i = 0$ y $\lambda_i = 1$).

Observación Si la consulta no tiene variables extremas (Definición 4.1) existencialmente cuantificadas, las restricciones en (5.3) son irrelevantes, debido a que todas las restricciones de la forma $R_{(w_i)_{i=1}^m}$ usarán sólo variables $X_{e=w}$.

5.3 Garantías con Eliminación de Variables Frecuentes

Existen casos en los cuales se tiene $\bar{t} \in \Box Q(\pi)$, y los algoritmos terminan en el primer paso. Esto se debe a que si $\bar{t} \in \Box Q(\pi)$ no hay solución entera para $(\mathcal{P})$, pero además existen ocasiones en que tampoco hay solución real. En aquellos casos en que $\bar{t} \in \Box Q(\pi)$ y no hay solución real para $(\mathcal{P})$ nuestros algoritmos terminan su ejecución en el primer paso al resolver el problema relajado. En esta sección se buscará caracterizar algunos de esos casos. Se establecerá el caso donde existe un subconjunto de restricciones R que usa todas las posibles variables relacionadas a los arcos involucrados en las restricciones de R. Si las diferentes variables no son demasiadas, como todas ellas juntas tienen que sumar la cantidad de arcos que hay en las restricciones de R, el conjunto de restricciones R es infactible.

Para establecer el caso que se está buscando, se necesita el siguiente algoritmo que encuentra el menor conjunto de variables que se puede remover de las variables en las restricciones de R, de forma de remover al menos una variable de cada restricción en R.

Algoritmo 3 Eliminación de Variables Frecuentes EVF

Require: R un conjunto de restricciones

$S \leftarrow \{X_1, ..., X_n\}$ el conjunto de variables que aparece en el menos 2 restricciones de R
if $S = \phi$ **then**
    return V un conjunto formado por una variable de cada restricción
else
    for $j = 1 \rightarrow n$ **do**
        $R_j \leftarrow \{r_1, ..r_{k_j}\}$ el conjunto de restricciones de R en que no aparece x_j
        $V_j \leftarrow EVF(R_j) \bigcup \{x_j\}$
    end for
    return V el V_j con menos variables
end if

Es claro por inducción que con el *Algoritmo* 3 se obtiene EVF(R) el menor conjunto de variables que son necesarias de remover en las restricciones de R para lograr eliminar al menos una variable de cada restricción.

Siguiendo la notación del *Algoritmo* 3, n será la cantidad de variables en S, donde S será el conjunto de variables que aparecen en al menos 2 restricciones de R. Hay que notar que n está acotado por la cantidad de restricciones ($|R|$) multiplicada por el número de variables en las restricciones, el cual está acotado por la consulta. Luego $n \in \mathcal{O}(|R|)$ lo cual es a lo más polinomial en el tamaño de π. Por lo tanto se puede tardar a lo más tiempo exponencial en computar EVF(R), más adelante en esta sección se mostrará que esto no presenta un problema para poder computar Certain Answers.

El siguiente Teorema muestra como usar el *Algoritmo* 3 EVF para encontrar clases donde se pueda resolver el problema de Certain Answers en tiempo polinomial con *Algoritmo* 1 y *Algoritmo* 2, pero antes son necesarias algunas definiciones.

Definición 5.6. Dados $Q, \pi, \bar{t}$ como en el Teorema 4.2, ($\mathcal{P}$) y ($\mathcal{P}'$) las representaciones como problemas de programación lineal del problema de Certain Answers y un conjunto de restricciones R de las restricciones $R_{(w_i)_{i=1}^m}$, sea $E(R)$ el conjunto de arcos tal que $e \in E(R)$ si y sólo si existe algún $X_{e=w}$ en alguna de las restricciones de R. Definimos

$$V_{\text{EVF}}(R) = \{X_{e=w} : e \in E(R), w \in L_e\} \setminus \text{EVF}(R).$$

Escribimos V_{EVF} si R es claro del contexto. Decimos que la *Condición de Eliminación de Variables en* R se satisface si y sólo si:

1. Las restricciones en R sólo tienen variables de la forma $X_{e=w}$
2. Para cada $e \in E(R)$, se tiene que $\forall w \in L_e$ la variable $X_{e=w}$ aparece en alguna de las restricciones de R.
3. $|V_{\text{EVF}}| < |E(R)|$

Teorema 5.7. *Dados $Q, \pi, \bar{t}$ como en el Teorema 4.2, $(\mathcal{P})$ y $(\mathcal{P}')$ las representaciones como problemas de programación lineal del problema de Certain Answers, considerando las siguientes restricciones:*

$$\forall (w_i)_{i=1}^m \in \prod_{i=1}^m L_i \quad R_{(w_i)_{i=1}^m}$$

si existe un subconjunto de restricciones R tal que la Condición de Eliminación de Variables en R *se satisface, entonces $\bar{t} \in \Box Q(\pi)$. Es más* Algoritmo 1 *y* Algoritmo 2 *responden correctamente en tiempo polinomial.*

Demostración. Como $\text{EVF}(R)$ es el menor conjunto de variables que es necesario quitar de las variables en las restricciones de R para eliminar al menos una variable de cada restricción en R, entonces V_{EVF} es el conjunto más grande de variables de las restricciones en R que se obtiene eliminando al menos una variable de cada restricción.

Cómo todas las variables relacionadas a los arcos en $E(R)$ aparecen en las restricciones R, y por cada arco ellas deben sumar 1, entonces todas juntas deben sumar $E(R)$, pero si las variables en V_{EVF} son menos que $|E(R)|$ significa que se necesitan más variables para asignar valor positivo, por lo tanto incluso tomando una solución no entera, es necesario que algunas variables que fueron seleccionadas en $\text{EVF}(R)$ tengan valor positivo. Por lo tanto todas las restricciones tienen todas sus variables que no fueron seleccionadas en $\text{EVF}(R)$ (o sea las en V_{EVF}) con valor 1, debido a que $|E(R)| > |V_{\text{EVF}}|$.

Como es necesario asignar valor positivo a las variables en $\text{EVF}(R)$, existe al menos una de ellas x que debe tener valor positivo $x > 0$, luego por minimalidad del conjunto $\text{EVF}(R)$ debe existir al menos una restricción r en R tal que sólo se eliminó x de ella. Como r ya tenía todas sus demás variables con valor 1 y tiene a x con valor positivo se tiene que r no se satisface.

Entonces el problema relajado es infactible, luego los algoritmos responden con correctitud y en tiempo polinomial solucionando el problema relajado en el primer paso. □

Es interesante observar que *Condición de Eliminación de Variables en R* captura casos triviales causados por un grupo de restricciones que son infactibles entre ellas. Estos casos

pueden pasar desapercibidos, incluso siendo triviales, debido a múltiples razones. Ya sean cosas simples como que la GDB incompleta es demasiado grande y hay unos pocos arcos $E(R)$ en ella, donde la consulta puede ser evaluada en cada completación o cosas más complejas como que la GDB incompleta tenga una estructura poco común en la cual hayan escasas partes donde poder evaluar la consulta. Además, cabe destacar que no es necesario verificar la *Condición de Eliminación de Variables en R* (lo que puede tomar harto tiempo) para resolver el problema, sólo se necesita satisfacer esta condición.

Una versión más general del teorema anterior es el caso en el que existe π' una GDB incompleta, contenida en π en la cual se tiene $\bar{t} \in \Box Q(\pi')$.

Ejemplo 5.2. Sea $Q = (x, 1101010100, y)$ una CRPQ y $\pi = (V, E)$ una GDB incompleta, tal que $V = \{a_1, a_2, a_3, b, c, d_1, d_2, d_3\}$ y hay arcos e_i de a_i a b etiquetados 101, 1 y 10, un arco f de b a c etiquetado con el lenguaje $0|1$, arcos g_i de c a d_i etiquetados por 10100, 1010100 y 010100 como se muestra en la siguiente figura:

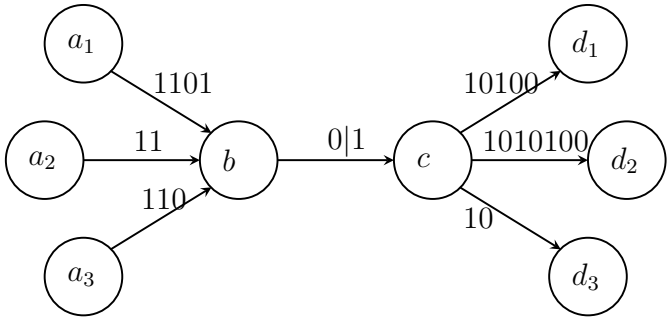

Figura 5.1: Ejemplo 5.2 GDB incompleta π

Las restricciones de la forma $R_{(w_i)_{i=1}^m}$ son:

$$\begin{aligned} X_{e_1=101} &+ X_{f=0} &+ X_{g_1=10100} &\leq 2 \\ X_{e_2=1} &+ X_{f=0} &+ X_{g_2=1010100} &\leq 2 \\ X_{e_3=10} &+ X_{f=1} &+ X_{g_1=010100} &\leq 2 \end{aligned}$$

Usando *Algoritmo* 3 EVF se obtienen los siguientes conjuntos:

$$\begin{aligned} EVF(R_{(w_i)_{i=1}^m}) &= \{X_{f=0}, X_{e_3=10}\} \\ V_{\text{EVF}}(R_{(w_i)_{i=1}^m}) &= \{X_{e_1=101}, X_{g_1=10100}, X_{e_2=1}, X_{g_2=1010100}, X_{f=1}, X_{g_1=010100}\} \end{aligned}$$

Se observa que en el lugar de $X_{e_3=10}$ puede estar cualquiera de las otras variables de la tercera ecuación. Entonces $|V_{\text{EVF}}| = 6 < 7 = |E(R)|$. Por lo tanto por el Lema 5.7 se obtiene $\bar{t} \in \Box Q(\pi)$.

Capítulo 6

Caso General

En los capítulos anteriores se utilizaron sólo CRPQs con lenguajes finitos. En este capítulo se muestra como establecer el problema de Certain Answers como un problema de programación lineal entera, quitando las restricciones de lenguajes finitos para la consulta. En el Lema 6.1 se muestra que a pesar de que los lenguajes en la consulta puedan ser infinitos es posible reducir el problema a un caso con lenguajes finitos. Luego en el Teorema 6.2 se prueba que es posible plantear el problema como un problema de programación lineal entera y se entrega un algoritmo para resolverlo. Finalmente en el Teorema 6.3 se muestra que una de las garantías de ejecución en tiempo polinomial expuestas en el capítulo anterior para el problema con CRPQs con lenguajes finitos también es válida en el caso general.

El siguiente lema muestra que se puede evitar el problema de lenguajes infinitos en la consulta, pero cambiándolos por lenguajes finitos de tamaño polinomial en el tamaño de la GDB incompleta. Ese cambio dificulta encontrar buenos algoritmos de aproximación, debido a que en $(\mathcal{P})$ el número de variables y el de restricciones son exponenciales en el tamaño de los lenguajes en las RPQs de la consulta. Por lo tanto en la construcción de $(\mathcal{P})$ se obtiene un problema de programación lineal entera equivalente al problema de Certain Answers, pero esta vez con un número exponencial de de variables y restricciones.

Lema 6.1. *Dada una CRPQ* $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$*, una base de datos incompleta* π *y una tupla* $\bar{t}$ *de la misma aridad que* Q*. Existe una CRPQ* $Q' = \bigwedge_{i=1}^{m}(x_i, \bar{L}_i, y_i)$ *tal que cada* $\bar{L}_i$ *es finito,* $|\bar{L}_i| \in \mathcal{O}(|\pi|)$ *y:*

$$\bar{t} \in \Box Q(\pi) \Leftrightarrow \bar{t} \in \Box Q'(\pi)$$

Demostración. En la demostración de la Proposición 4 de [1] se establece que si se tiene una consulta tal que cada variable en Q aparece en π (se tiene debido a que $\bar{t}$ es una tupla de nodos de π) y hay una GDB $G \in Rep(\pi)$ tal que $Q(G) = F$, entonces existe una GDB $G' \in Rep(\pi)$ tal que $Q(G') = F$. La misma proposición establece que el largo de cada camino en G'asociado a un arco de π está acotado por una constante c que depende de la formula

lógica de segundo orden monádica de Q y linealmente en el tamaño de los lenguajes de π (entonces depende linealmente del tamaño de π), y se puede encontrar G'. Por lo tanto es necesario verificar sólo las GDBs $G \in Rep(\pi)$ tal que sus caminos tienen tamaño en $\mathcal{O}(|\pi|)$ para poder determinar si $\bar{t} \in \Box Q(\pi)$.

Si $\bar{t} \in \Box Q(\pi)$ existe $G \in Rep(\pi)$ y $Q(G) = V$. Como se vio, se puede suponer que el tamaño de G está en $\mathcal{O}(|\pi|)$, luego para cada i existe una palabra w_i en L_i tal que es posible evaluar (x_i, w_i, y_i) entre los nodos a y b en G. Se considera el autómata $\mathcal{A}_G$ que sus nodos son los nodos en G, su función de transición es los arcos en G, su nodo de inicio es a y su nodo de término es b. Sea L_G el lenguaje regular de este autómata, luego $w_i \in L_i \cap L_G = L'_i$. El tamaño del autómata que reconoce L'_i es el tamaño del autómata que reconoce L_G ($|G|$) por el tamaño del autómata que reconoce L_i ($\mathcal{O}(|Q|)$). Como $\mathcal{O}(|Q|)$ es constante, el tamaño del autómata que reconoce L'_i es $D|G|$ donde D es una constante que depende de Q.

Luego el tamaño de la expresión regular de L'_i está acotado por $p(|G|)$ donde p es el polinomio de tiempo que toma la transformación de AFD a expresión regular (con la constante D incluida), así se puede encontrar una palabra $w'_i \in L'_i$ tal que $|w'_i| \leq p(|G|)$, debido a que la expresión regular tiene la misma cota. Además el tamaño de G está acotado por una constante c por $|\pi|$, por lo tanto hay que buscar palabras en L_i cuyo tamaño está acotado por $p(c|\pi|)$.

Entonces para todo i se puede construir el lenguaje $\bar{L}_i$ con las palabras en el lenguaje L_i cuyo largo es menor que $p(c|\pi|)$. □

El resultado anterior es útil porque permite cambiar una consulta $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ donde los lenguajes L_i pueden tener un número infinito de palabras, por una consulta $Q' = \bigwedge_{i=1}^{m}(x_i, \bar{L}_i, y_i)$ donde los $\bar{L}_i$ son todos finitos. Sin embargo, el tamaño de los lenguajes $\bar{L}_i$ es polinomial en π, lo que vuelve menos útil a la reducción en el tamaño de los lenguajes en los arcos de π. Debido a que se obtiene para π una reducción a una GDB incompleta con lenguajes de un número exponencial de palabras en los arcos.

Esto es un problema porque si π tiene al menos un lenguaje con un número exponencial de palabras, entonces se tiene un número exponencial de variables en la representación como un problema de programación lineal entera ($\mathcal{P}$). Es más, incluso si se tuviese una cantidad finita, polinomial de variables (cada lenguaje en los arcos de π necesitaría tener un tamaño polinomial), se tendrá una cantidad exponencial de restricciones, puesto que se toman combinaciones de las palabras en la consulta (las que son polinomiales en π) y luego particiones de las palabras (cantidad exponencial). Además para cada partición de palabras de la consulta, como las palabras pueden tener largo polinomial en π, se tiene una cantidad exponencial de combinaciones de nodos en π asociados a la partición.

Los argumentos recién expuestos muestran lo complejo que es tratar de resolver el problema general mediante su versión de programación lineal entera. A pesar de esto, existe una

técnica que se puede utilizar para abordarlo. Lo que se puede hacer es usar generación de variables y generación de restricciones [25] en la versión del problema $(\mathcal{P})$.

Teorema 6.2. *Dada una CRPQ* $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$*, una GDB incompleta* π*, una tupla de nodos* $\bar{t}$ *de la misma aridad que* Q*, el problema de determinar si* $\bar{t} \in \Box Q(\pi)$ *se puede establecer como un problema de programación lineal entera* $(\mathcal{P})$*, de forma que:*

$$\bar{t} \notin \Box Q(\pi) \Leftrightarrow (\mathcal{P}) \text{ es factible}$$

y existe un algoritmo que lo resuelve.

Demostración. El Lema 6.1 asegura que se puede suponer que para cada i el lenguaje L_i es finito con tamaño polinomial en $|\pi|$. Como se vio en los capítulos anteriores se puede construir un problema de programación lineal entera $(\mathcal{P})$ equivalente al problema de Certain Answers, pero esta vez con un número exponencial de variables y restricciones. Entonces no se puede construir completo $(\mathcal{P})$ en un tiempo polinomial, por lo tanto sólo se construirán partes de él en el siguiente algoritmo:

1. Se toman algunas variables, al menos una por cada arco en π, se toman algunas de las restricciones de la forma $R_{(w_i)_{i=1}^m}$ (eventualmente podría ser ninguna), y se toman las restricciones $\forall e \in E \sum_{w \in L_e} X_{e=w} = 1$ solamente con las variables incluidas.

2. Se resuelve el problema relajado.

3. Si el problema es infactible, y no se pueden agregar más variables asociadas a los arcos involucrados en las restricciones que no se satisfacen, entonces el problema es efectivamente infactible y se tiene $\bar{t} \in \Box Q(\pi)$.

4. Si el problema es infactible, y se pueden agregar más variables asociadas a los arcos involucrados en las restricciones que no se satisfacen, se agrega una variable por cada arco involucrado y se regresa al punto 2.

5. Si se obtiene una solución no entera se puede bifurcar entre los valores de alguna de las variables no enteras e ir al paso 2.

6. Si se obtiene una solución entera, se toma la GDB asociada $G \in Rep(\pi)$ y se evalúa la consulta en ella. Si $\bar{t} \notin Q(G)$, se tiene $\bar{t} \notin \Box Q(\pi)$, pero si $\bar{t} \in Q(G)$, significa que existe una restricción insatisfecha que no se ha agregado aún. La cual es la restricción asociada a la evaluación de la consulta en G. Luego se pueden tomar palabras $w_i \in L_i$ que hagan posible la evaluación de la consulta en G, agregar las restricciones $R_{(w_i)_{i=1}^m}$ y volver al paso 2.

□

Observación El principal problema es que el algoritmo puede tomar tiempo exponencial por la cantidad de variables que sean necesarias de agregar, y nuevamente tiempo exponencial por las bifurcaciones. Pero lo bueno es que ese es el peor caso, y si se maneja el problema en específico que se quiere tratar se pueden encontrar buenas maneras de escoger que variables y que restricciones agregar.

Una interrogante que hasta ahora no se ha considerado es si se pueden aplicar los resultados del Capítulo 5 en el problema general. La respuesta es que no se puede aplicar el Teorema 5.1 porque requiere $Q \in \mathcal{F}$. Se puede aplicar el Teorema 5.3, pero para resolver el problema con los algoritmos del Capítulo 4 es necesario revisar todas las restricciones, e incluso el problema relajado puede tardar tiempo exponencial por el número de variables. Finalmente, se puede aplicar el Teorema 5.7 debido a que sólo requiere subconjuntos de restricciones.

Teorema 6.3. *Dados $Q, \pi, \bar{t}$ y $(\mathcal{P})$ como en el Teorema 6.2, se consideran las siguientes restricciones:*

$$\forall (w_i)_{i=1}^m \in \prod_{i=1}^m L_i \quad R_{(w_i)_{i=1}^m}$$

Si existe un subconjunto de restricciones R tal que la Condición de Eliminación de Variables en R *se satisface, entonces $\bar{t} \in \Box Q(\pi)$ y el algoritmo de aproximación del Teorema 6.2 responde con correctitud en tiempo polinomial, usando las restricciones en R, en el paso 1.*

Demostración. La demostración es exactamente la misma a la del Teorema 5.7 porque R es sólo un subconjunto de restricciones. □

Capítulo 7

Conclusiones

A lo largo de esta tesis hemos visto que determinar cuando $\bar{t} \in \Box Q(\pi)$ es verdadero o no ha probado ser una difícil tarea. Pero hay casos en los cuales el problema resulta ser posible de resolver en tiempo polinomial.

Si no se tiene alguna de las condiciones $(C1)$, $(C2)$ o $(C3)$ establecidas en el Teorema 3.9, el problema es coNP-completo, pero se demostró de dos formas distintas que si se tienen las tres condiciones, el problema puede ser resuelto en tiempo polinomial. Se mostró que hay otros casos en los que también se puede resolver el problema en tiempo polinomial, como cuando en la formulación de programación lineal del problema existe un grupo de restricciones R que satisfacen la *Condición de Eliminación de Variables en* R, o condiciones tan simples como la total unimodularidad de la matriz de restricciones.

Es muy importante notar que la *Condición de Eliminación de Variables en* R captura casos triviales causados por un grupo de restricciones que son infactibles entre ellas. Esos casos podrían no ser vistos fácilmente, incluso siendo triviales. Lo que se puede deber a un sin fin de razones, o a cosas simples como que la GDB incompleta es muy grande y hay una pequeña parte de ella en la que se puede evaluar la consulta en cada completación.

Además, se mostró varias maneras de obtener algoritmos que terminan en tiempo polinomial bajo algunas condiciones, sin necesidad de verificar las condiciones de antemano. Es más, en otros casos, es posible detener el algoritmo antes de terminar o escoger su versión heurística para obtener un candidato a solución.

Sin embargo, más importante que las condiciones para las cuales se obtienen soluciones en tiempo polinomial, se establecieron maneras de ver el problema como un problema de programación lineal entera. Esto es una poderosa herramienta para encontrar nuevas condiciones para resolver el problema, usando las bien conocidas técnicas para resolver problemas de programación lineal.

Finalmente, se estudió el caso general del problema, donde los lenguajes en las RPQs de la consulta no son de tamaño finito. Se mostró un algoritmo para resolver la formulación de programación lineal entera del problema de Certain Answers en el caso general. A pesar de que el algoritmo puede tardar tiempo exponencial, es una muy buena forma de obtener un acercamiento a una solución en tiempo polinomial al problema. Además la *Condición de Eliminación de Variables en R* es válida para el algoritmo.

Bibliografía

[1] P. Barceló, L. Libkin, J. Reutter. *Querying Graph Patterns. In PODS, 2011.*

[2] R. Bosch, M. Trick. *Introductory Tutorials in Optimization and Decision Support Techniques, 2007.*

[3] I. Cruz, A. Mendelzon, P. Wood. *A graphical query language supporting recursion. In SIGMOD, 1987.*

[4] D. Calvanese, G. de Giacomo, M. Lenzerini, M. Y. Vardi. *Rewriting of regular expressions and regular path queries. JCSS, 2002.*

[5] S. Abiteboul, P. Buneman, D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML. Morgan Kauffman, 1999.*

[6] M. P. Consens, A. Mendelzon. *GraphLog: a visual formalism for real life recursion. In PODS, 1990.*

[7] D. Calvanese, G. de Giacomo, M. Lenzerini, M. Y. Vardi. *Containment of conjunctive regular path queries with inverse. In KR, 2000.*

[8] R. Fagin, P. Kolaitis, R. Miller. *Data Exchange: Semantics and Query Answering, 2003.*

[9] M. Vardi. *The Complexity of Relational Query Languages. In PODS, 1982.*

[10] S. Cohen, Y. Sagiv. *An abstract framework for generating maximal answers to queries. In ICDT, 2005.*

[11] F. Olken. *Graph data management for molecular biology. OMICS, 2003.*

[12] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, U. Alon. *Network motifs: simple building blocks of complex networks. Science 298(5594), 2002.*

[13] U. Leser. *A query language for biological networks. Bioinformatics 21, 2005.*

[14] R. Ronen, O. Shmueli. *SoQL: a language for querying and creating data in social networks. In ICDE, 2009.*

[15] M. San Martin C. Gutierrez. *Representing, querying and transforming social networks with RDF/SPARQL. In ESWC, 2009.*

[16] C. Gutierrez, C. Hurtado, A. Mendelzon. *Foundations of semantic web databases. In PODS, 2004.*

[17] J. Perez, M. Arenas, C. Gutierrez. *Semantics and complexity of SPARQL. ACM TODS, 2009.*

[18] R. Angles, C. Gutierrez. *Survey of graph database models. ACM Comput. Surv, 2008.*

[19] M. Arenas, P. Barceló, L. Libkin, F. Murlak. *Relational and XML Data Exchange. Morgan & Claypool, 2010.*

[20] M. Lenzerini. *Data integration: a theoretical perspective. In PODS, 2002.*

[21] P. Barcelo, L. Libkin, A. Poggi, C. Sirangelo. *XML with Incomplete Information. Journal of the ACM, Vol. V. In PODS, 2009.*

[22] T. Imielinski, W. Lipski, Jr. *XML with Incomplete Information. Incomplete Information in Relational Databases. Journal of the ACM, 1984.*

[23] C. David, L. Libkin, F. Murlak. *Certain Answers for XML Queries. In PODS, 2010.*

[24] S. Abiteboul, R. Hull, V. Vianu. *Foundations of Databases. Addison-Wesley, 1995.*

[25] G. Desaulniers, J. Desrosiers, M. Solomon. *Column Generation, 1995.*

Apéndice del Capítulo 3

Intuitivamente incluso si la GDB incompleta tiene lenguajes de una infinita cantidad de palabras, no es necesario revisarlas todas. Debido a que la consulta es finita, sólo se necesita usar las palabras en la base de datos tal que tienen algo en común con las palabras en la consulta. Consideremos el siguiente ejemplo:

Ejemplo 8.1. Sea $Q = (x, L_1, y)$ con $L_1 = \{101, 1011\}$ y π una GDB incompleta con un arco e etiquetado por $L = 0^+$. Claramente es posible usar la palabra $w = 0$ para evaluar Q, pero no se puede usar ninguna de las palabras en $L \setminus \{0\}$ al evaluar Q (usando la palabra en la evaluación). Luego si la consulta es evaluada en π usando el arco e, será usando la palabra $w = 0$. Como las palabras en L_1 de Q no pueden ser evaluados a través del arco e con una palabra distinta de w, no es necesario usarlas todas, sólo es necesario usar una palabra (w). Además, es necesario usar w, pero también es necesario usar otra palabra para representar la posibilidad de escoger una palabra que impida evaluar la consulta mediante el uso del arco e. Por lo tanto se puede reducir el lenguaje L a $L' = \{0, 00\}$ y la solución al problema de Certain Answers seguirá siendo la misma.

Demostración de la Proposición 3.8

Demostración. Una forma más simple de ver esta proposición es viendo Q como un grafo, donde las variables en $\bar{x}$ e $\bar{y}$ son nodos, y para cada i la RPQ (x_i, L_i, y_i) es un arco entre x_i e y_i etiquetado por L_i. Aprovechando esto se utilizará más adelante la noción de *camino* en la consulta, donde una camino c es un conjunto de RPQs de Q que están conectadas, esto es $c = ((x_{i_j}, L_{i_j}, y_{i_j}))_{j=1}^{l}$ tal que $\forall j\ 1 \leq j \leq l-1\ y_{i_j} = x_{i_{j+1}}$.

Es necesario definir los diferentes tipos de palabras que son relevantes para un lenguaje, esto es, clasificar las palabras según que palabras del lenguaje son prefijos, sufijos o subpalabras y cuales no. Luego será necesario identificar los tipos de palabras relevantes para una CRPQ.

Definición 8.1. Dado un conjunto finito de palabras L y una palabra w, se dirá que w es de *L-tipo* (W_1, W_2, W_3), donde $W_1, W_2, W_3 \subseteq L$, ssi:

$$w \in \bigcap_{\omega \in W_1} \alpha\Sigma^* \bigcap_{\alpha \in W_2} \Sigma^*\alpha\Sigma^* \bigcap_{\alpha \in W_3} \Sigma^*\alpha \bigcap_{\alpha \in L \setminus W_1} (\alpha\Sigma^*)^c \bigcap_{\alpha \in L \setminus W_2} (\Sigma^*\alpha\Sigma^*)^c \bigcap_{\alpha \in L \setminus W_3} (\Sigma^*\alpha)^c.$$

Los complementos de W_1, W_2 y W_3 son tomados con respecto a L. Intuitivamente el conjunto W_1 es el conjunto de prefijos, W_2 es el conjunto de subpalabras y W_3 es el conjunto de sufijos de w que están en L. Es importante notar que calcular el $L-tipo$ de una palabra toma tiempo polinomial en el tamaño de L, debido a que sólo se necesita revisar cada palabra en L.

Ejemplo 8.2. Sea $L = 10^*1|000$ un lenguaje regular, luego la palabra $w_1 = 100100$ tiene $L - tipo$ $(1001, 1001, \phi)$ y la palabra $w_2 = 010001$ tiene $L - tipo$ $(\phi, 10001|000, 10001)$.

Para extender esta idea a CRPQs, sólo se necesita usar un lenguaje L de cada palabra que es posible de usar en la consulta. Los ciclos en la consulta generan un pequeño problema, puesto que al buscar cualquier palabra que se pueda usar en la consulta hay que hacerlo revisando las palabras que se pueden formar en la consulta a través de varias RPQs conectadas. Lo que se hará es considerar todas las palabras que se puedan formar en la consulta sin repetir RPQs, debido a que al evaluar la consulta en el patrón no es necesario repetir arcos.

Definición 8.2. Dada una CRPQ $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ con L_i un lenguaje finito $\forall i$, se toma C el conjunto de caminos de tamaño a lo más m en Q, y:

$$L' = \bigcup_{c=(e_1,\ldots,e_k)\in C} L_1 \ldots L_k$$

La cantidad de caminos en C está acotada por el tamaño de Q (por 2^m). Se considera L el conjunto de subpalabras de las palabras en L'. Se dice que para $W_1, W_2, W_3 \subseteq L$ una palabra w es de *Q-tipo* (W_1, W_2, W_3) si w es de $L - tipo$ (W_1, W_2, W_3).

Es importante notar que dada una CRPQ Q, la cantidad de diferentes *Q-tipos* está acotad en el tamaño de L, el cual está acotado en el tamaño de Q (y de sus lenguajes).

Ahora que se tienen las definiciones necesarias se puede continuar con la demostración de la Proposición 3.8.

Como los lenguajes en Q son finitos, el número de diferentes *Q-tipos* también es finito. Entonces, para construir π' se pueden cambiar los lenguajes en π por lenguajes finitos con los mismos diferentes *Q-tipos* que los que hay en los lenguajes en π (en cada arco, arco por arco). para verificar si en un lenguaje L hay palabras de un *Q-tipo* (W_1, W_2, W_3) basta con intersectar L con el lenguajes:

$$\bigcap_{\omega\in W_1} \alpha\Sigma^* \bigcap_{\alpha\in W_2} \Sigma^*\alpha\Sigma^* \bigcap_{\alpha\in W_3} \Sigma^*\alpha \bigcap_{\alpha\in \bar{L}\setminus W_1} (\alpha\Sigma^*)^c \bigcap_{\alpha\in \bar{L}\setminus W_2} (\Sigma^*\alpha\Sigma^*)^c \bigcap_{\alpha\in \bar{L}\setminus W_3} (\Sigma^*\alpha)^c$$

donde $\bar{L}$ es el conjunto L de la Definición 8.2. Lo que se puede hacer en tiempo finito porque Q tiene una cantidad finita de palabras. Luego se puede verificar la presencia de palabras de cada *Q-tipo* en los lenguajes de L en tiempo polinomial en L, para escoger

las palabras sólo se necesita escoger cualquier palabra en la intersección que no sea vacía. Luego el tiempo total que tarda la construcción de π' es la suma de los tiempos que toma intersectar cada lenguaje de arco con el lenguaje L de la definición de *Q-tipo*, luego tarda tiempo polinomial en el tamaño de π. Y claramente se tiene la equivalencia de las consultas por el Lema 8.3. □

Lema 8.3. *Sea $\pi = (V, E)$ una GDB incompleta y $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ una CRPQ sobre π, se considera el arco $e \in E$ y su lenguaje asociado L_e, la palabra $w \in L_e$ con* Q-tipo *(W_1, W_2, W_3), si se considera otra GDB incompleta $\pi' = (V, E')$ exactamente igual a π, pero cambiando la palabra w por una palabra u en L_e en el arco $e \in E'$ del mismo* Q-tipo*, se obtiene:*

$$\bar{t} \in \Box Q(\pi) \Leftrightarrow \bar{t} \in \Box Q(\pi') \quad \forall \bar{t}$$

Demostración. Sea t una tupla de la misma aridad que Q, si $t \in Q(\pi)$, entonces existe una base de datos $G \in Rep(\pi)$ tal que es posible evaluar la consulta Q en G asignando los nodos libres de Q a t. Si G no usa la palabra $w \in L_e$, se tiene $G \in Rep(\pi')$. Si G usa la palabra $w \in L_e$, hay al menos un camino en Q que necesitan la presencia de w en el arco e de G para evaluar el camino de la consulta. Sea $c = ((x_{i_j}, L_{i_j}, y_{i_j}))_{j=1}^{k}$ este camino, y $v = v_1...v_k$ la palabra que se lee en él.

Es importante notar que v está en el conjunto L de la definición de *Q-tipo* porque se puede asumir que el camino no repite arcos, si lo hiciera, sólo sería necesario revisar el ciclo una sola vez. Como v necesita a w en el arco e significa que v ocupa entero a w, un sufijo de v está al comienzo de w, un prefijo de v está al final de w o w se lee entero en v, pero lo mismo ocurre con u debido a que w y u tienen el mismo *Q-tipo* (para cualquier subpalabra de v; u y w ambos la tienen o no juntos como un prefijo, sufijo o subpalabra). Así, si se cambia w por u, v se evalúa de la misma forma. Luego $t \in \Box Q(\pi')$, y se tiene el converso por simetría de la demostración. □

Apéndice del Capítulo 4

En este apéndice se presentan algunas reducciones para el problema de Certain Answers que facilitan las demostraciones en el Capítulo 4, mostrando que se pueden considerar casos simples para probar casos más difíciles. Se presentan algunas reducciones al problema de tener nodos existencialmente cuantificados en la consulta.

La siguiente proposición muestra que una CRPQ Q, puede ser transformada en una CRPQ Q' equivalente a Q, eliminando sus variables existencialemente cuantificadas que aparecen una sola vez al comienzo de una RPQ de Q y una sola vez al final de una RPQ de Q.

Proposición 8.4. *Sea $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ una CRPQ, sea N_l el conjunto de las variables libres y las variables fijas en Q y N_e el conjunto de las variables existencialmente cuantificadas de Q. Sea*

$$W = \{n \in N_e : \exists! i_1 \quad n = y_{i_1} \wedge \exists! i_2 \quad n = x_{i_2}, \quad i_1 \neq i_2\},$$

luego existe una CRPQ $Q' = \bigwedge_{i=1}^{k}(x'_i, L'_i, y'_i)$ con $k \leq m$, tal que el su conjunto de variables libres y variables fijas es también N_l y el conjunto de variables existencialmente cuantificadas es $N_e \setminus W$ tal que para una tupla $\bar{t}$ de ids de nodos de la misma aridad que Q, y para toda GDB incompleta π se tiene:

$$\bar{t} \in \Box Q(\pi) \Leftrightarrow \bar{t} \in \Box Q'(\pi).$$

Demostración. Sea $Q = \bigwedge_{i=1}^{m}(x_i, L_i, y_i)$ una CRPQ, sea π una GDB incompleta y $\bar{t}$ una tupla de ids de nodos de la misma aridad que Q. Sea $n \in W$, entonces $\exists! i_1 \; n = y_{i_1} \wedge \exists! i_2 \; n = x_{i_2}$, se puede crear la CRPQ Q' igual a Q, pero reemplazando $(x_{i_1}, L_{i_1}, y_{i_1})$ y $(x_{i_2}, L_{i_2}, y_{i_2})$ por $(x_{i_1}, L_{i_1}L_{i_2}, y_{i_2})$ renombrado como $(x_{i^*}, L_{i^*}, y_{i^*})$.

Si $\bar{t} \in \Box Q(\pi)$, sea $G = (N_G, V_G) \in Rep(\pi)$, por lo tanto $\bar{t} \in Q(G)$, y luego existe $n_x, n_y \in N_G$ tal que al evaluar Q sobre G los nodos x_{i_1} e y_{i_2} son mapeados a n_x y n_y respectivamente.

Como $\bar{t} \in \Box Q(G)$, existen ρ_1 y ρ_2 caminos en G etiquetados por $w_1 \in L_{i_1}$ y $w_2 \in L_{i_2}$ respectivamente y un nodo $\bar{n}$ en G tal que el nodo final de ρ_1 es $\bar{n}$, $\bar{n}$ es el nodo inicial de ρ_2 y es posible evaluar Q mapeando n a $\bar{n}$. Luego la RPQ $(x, L_{i_1}L_{i_2}, y)$ puede ser evaluada

en G en el camino $\rho_1\rho_2$. Por lo tanto usando el mapeo que permite evaluar Q en G, también se puede evaluar $(x_{i^*}, L_{i^*}, y_{i^*})$ de una forma consistente con el resto de las variables en la consulta Q'. Luego se tiene $\bar{t} \in \Box Q'(G)$ y $\bar{t} \in \Box Q'(\pi)$.

Ahora se necesita probar que si $\bar{t} \in \Box Q'(\pi)$ con la misma Q' que se construyó recién se tendrá $\bar{t} \in \Box Q(\pi)$.

Si $\bar{t} \in \Box Q'(\pi)$, sea $G = (N_G, V_G) \in Rep(\pi)$, por lo tanto $\bar{t} \in Q'(G)$ y es posible evaluar $(x_{i^*}, L_{i^*}, y_{i^*})$ sobre G. Luego existe un camino ρ entre los nodos $n_x, n_y \in N_G$ tal que al evaluar Q en G los nodos x_{i^*} e y_{i^*} son mapeados a n_x y n_y respectivamente. Es más, el etiquetado del camino ρ es una palabra $w \in L_{i^*} = L_{i_1}L_{i_2}$, luego es posible separar la palabra w en $w = w_1w_2$ donde $w_1 \in L_{i_1}$ y $w_1 \in L_{i_2}$.

Como ρ es un camino entre n_x y n_y etiquetado por w_1w_2 luego existe un nodo $\bar{n}$ en G y caminos ρ_1 y ρ_2 en G etiquetados por $w_1 \in L_{i_1}$ y $w_2 \in L_{i_2}$ tal que ρ_1 termina en $\bar{n}$ y ρ_2 empieza en $\bar{n}$. Luego las RPQs $(x_{i_1}, L_{i_1}, y_{i_1})$ y $(x_{i_2}, L_{i_2}, y_{i_2})$ pueden ser evaluadas en G. Es más, con el mismo mapeo de las variables de Q', es posible mapear las variables de Q y evaluarla mapeando n a $\bar{n}$. Luego $\bar{t} \in Q'(G)$ y por lo tanto $\bar{t} \in \Box Q(\pi)$.

Hay que notar que el conjunto W que se puede crear de Q' es el mismo que se crea de Q, pero con un elemento menos (n) que el conjunto W que se crea de Q. Luego por indicción se puede crear la CRPQ Q' del enunciado tal que $\bar{t} \in \Box Q'(\pi)$ si y sólo si $\bar{t} \in \Box Q(\pi)$. □

Es importante observar que si la consulta Q original tiene lenguajes finitos, la consulta resultante Q' también tendrá lenguajes finitos.