Utilización de Extensiones de Lógica de Primer Orden para la Computación de Queries en problemas de Redes

Gagliardi, Edilma Olinda
Maldocena, Paulino Daniel
Reyes, Nora Susana

{oli, pmaldo, nreyes}@unsl.edu.ar

Universidad Nacional de San Luis
Facultad de Ciencias Físico, Matemáticas y Naturales
Departamento de Informática
Ejército de Los Andes 950
San Luis

ABSTRACT

Este trabajo presenta resoluciones de queries a una base de datos que mantiene información acerca de redes de computadoras, utilizando Lógica de Primer Orden extendida con los cuantificadores Clausura Transitiva, Clausura Transitiva Determinística, Puntos fijos en sus variantes Inflaccionario y No Inflaccionario, en su versión Determinística.

Para los queries mostrados no alcanza con Lógica de Primer Orden, mientras que con las extensiones planteadas es posible expresarlos. En algunos casos, ellos representan casos críticos y pertenecen a una clase de consultas no expresables en lenguajes cuya expresividad sea equivalente al de la Lógica de Primer Orden, por lo que es necesario contar con otros cuantificadores de mayor poder expresivo.

La aplicación elegida es sólo relevante para observar la semántica de ciertos queries, los cuales son razonables de plantear sobre una base de datos, y cuya complejidad supera la expresividad de los lenguajes que se puedan tener disponibles para la resolución de los mismos.

Palabras Claves: Bases de Datos Relacionales; Queries; Lógica de Primer Orden; Cuantificadores de Clausura Transitiva y Punto Fijo; Teoría de Modelos Finitos.

1. INTRODUCCIÓN

Por medio de una *Base de Datos* obtenemos una representación simbólica de una realidad acotada. Para tal representación utilizaremos el *Modelo Relacional*. Como es deseable obtener información de la base de datos, es necesario contar con un lenguaje de consultas. Para nuestro modelo consideramos el *Álgebra Relacional (AR)* y el *Cálculo Relacional (CR),* ambos lenguajes son equivalentes en su poder expresivo, y más aún, equivalentes a la *Lógica de Primer Orden (FO)* [ULLMAN 88][ABITEBOUL 95b]. En este trabajo utilizaremos *FO* para la expresión de consultas a las bases de datos, dado que su poder expresivo es equivalente a los otros dos lenguajes relacionales.

También presentaremos la definición de la clase de consultas computables, llamada *Computable Queries* (*CQ*) [CHANDRA 80]. Para ello observamos el *Modelo Relacional* en el marco de la *Teoría de Modelos Finitos* [EBBINGHAUS 95], tal que el esquema de la Base de Datos Relacional está definido por un vocabulario relacional finito σ, y la instancia de Base de Datos es una σ-estructura, donde cada relación es de la aridad correspondiente para cada símbolo de σ, definida en un dominio finito dado. Así se definen *Clases de Estructuras Relacionales Finitas* para Modelos Relacionales, y se definen sus instancias como *Estructuras Relacionales Finitas*. Entonces, desde este punto de vista, vemos un query como una función cuyo dominio es el conjunto de las estructuras relacionales finitas de un cierto vocabulario, y cuyo codominio es el conjunto de las relaciones definidas en el dominio de la estructura relacional finita correspondiente para alguna aridad, q: $E_{\sigma,\, fin} \rightarrow E_{\langle R\rangle}$. En [CHANDRA 80] se propone como definición de la clase de queries computables *CQ* a la clase de funciones definidas en las clases de estructuras relacionales finitas, para cada vocabulario relacional finito, tales que son funciones recursivas parciales en alguna codificación lineal de las estructuras, y preservan isomorfismos en ellas.

Está demostrado que *FO* (o *AR* y *CR*) computan una clase restringida de consultas respecto de la clase *CQ*.

En este trabajo se muestra cómo *FO* aumentado con cuantificadores incrementa su poder expresivo, permitiendo resolver más queries de la clase *CQ*, aunque no se alcanza a cubrir la clase completa. Se mencionan cuantificadores, aunque algunos autores los tratan como operadores de trabajo; nosotros utilizamos la primera terminología dado que nos movemos en el marco de las lógicas.

Tomamos un caso de aplicación, una base de datos para mantener información acerca de redes de computadoras, y analizamos tipos de queries que pueden ser expresados en *FO*, queries que pueden ser expresados únicamente en las extensiones de FO propuestas, mostrando la expresividad (o restricciones) en cada caso.

Utilizamos un lenguaje de consultas a bases de datos relacionales que computa queries expresables en *FO* extendido con los *cuantificadores Clausura Transitiva*, *Clausura Transitiva Determinística*, *Puntos fijos Determínistico en su variante Inflaccionario* y *No Inflaccionario*. Este lenguaje es un subconjunto del que ha sido desarrollado e implementado por nuestro grupo de trabajo, utilizándolo como lenguaje modelo para la expresión y evaluación de consultas a bases de datos relacionales.

2. FO, BASES DE DATOS Y QUERIES

Sea $A = \{ (\ ,\ ) ,\ ,\ ,\ \wedge\ ,\ \vee\ ,\ \neg\ ,\ \rightarrow\ ,\ \exists\ ,\ \forall\ ,\ x_1\ ,...,\ x_n\ ,...\}$ un alfabeto infinito numerable.

Una *signatura* o *vocabulario relacional* σ es una secuencia finita de símbolos de relación; $\sigma = < R_1,..., R_m >$. Asociado con cada símbolo de relación R_i está la *aridad* r_i, con $1 \leq i \leq m$.

Se define el *Lenguaje de Primer Orden*, con vocabulario σ, denotado L_σ , al lenguaje construido a partir del alfabeto infinito numerable ($A \cup \sigma$), según las reglas habituales de construcción sintáctica. Para la semántica de los lenguajes *FO* se utilizará la habitual semántica formal de Tarski.

Siempre nos referiremos a lenguajes con igualdad, aunque no haremos explícita la existencia del símbolo de relación de la igualdad en los vocabularios que consideraremos.

Emplearemos los conceptos de *variables libres* y *variables ligadas* en su significación habitual. Llamamos *sentencia* a una fórmula sin variables libres. Usaremos la notación $\varphi(x_1,..., x_r)$ para indicar que las variables libres en la fórmula φ son $x_1,..., x_r$.

Nosotros utilizaremos la Lógica desde la perspectiva de *Teoría de Modelos*.

Denotaremos con *FO* al lenguaje (o lógica) de Primer Orden, en un sentido general, es decir, prescindiendo de qué lenguaje específico de primer orden nos estamos ocupando. Lo que nos interesa destacar con esta simbología, es la expresividad o axiomatizabilidad del lenguaje.

Llamamos σ-*estructura* a una estructura que tiene un conjunto y tantas relaciones (de la aridad adecuada), definidas en ese conjunto, como símbolos de relación haya en σ. Sea $\sigma = \langle R_1,..., R_m \rangle$, entonces si *B* es una σ-estructura lo denotaremos de la siguiente manera:

$$B = \langle D^B, R_1{}^B,..., R_m{}^B \rangle$$

Llamamos *dominio* de *B* al conjunto D^B (*dom*(*B*)). Las relaciones $R_1{}^B,..., R_m{}^B$ están definidas en D^B.

Una σ-estructura *B* definida de esta manera es llamada *relacional*, por ser σ una *signatura relacional* y es *finita* si el conjunto *dom*(*B*) es finito.

Denotaremos con E_σ al conjunto de todas las σ-estructuras, de cualquier cardinal finito o infinito. Y denotaremos con $E_{\sigma, fin}$ al conjunto de todas las σ-estructuras finitas.

Diremos que *v* es una *valoración* en la σ-estructura *B*, si *v* es una función definida desde el conjunto de variables de individuo de L_σ en el conjunto D^B.

Una *interpretación I* consiste de una σ-estructura *B* y de una valoración *v* en la σ-estructura *B*. Entonces, la interpretación de una fórmula en L_σ consiste de una estructura, de signatura acorde con la del lenguaje en cuestión, y de una función de valoración que sustituye cada variable por un elemento en el dominio de la interpretación; de forma tal de obtener un enunciado o sentencia, del que puede conocerse su valor de verdad. En particular, una sentencia de L_σ es interpretada por una estructura de signatura σ; de modo que, sustituyendo cada símbolo de relación de la sentencia por la relación correspondiente en la estructura y evaluando la sentencia de la forma habitual, sabremos su valor de verdad [EBBINGHAUS 84].

La *relación de satisfacción* (⊨) hace precisa la noción de que una fórmula sea verdadera en una interpretación dada. Si σ es un vocabulario relacional finito, $\varphi \in L_\sigma$ es una fórmula con variables libres $x_1,x_2,...,x_r$, y *B* es una σ-estructura (con $|D^B| = n$) , se denotará con la siguiente expresión que la fórmula φ es verdadera si es interpretada en la estructura *B*, con el elemento $d_{s_j} \in D^B$ asignado a la variable libre x_j, para $1 \leq s_j \leq n$, $1 \leq j \leq r : B \vDash \varphi(x_1,\ldots,x_r)[\, d_{s_1},...,d_{s_r}\,]$.

Consultas o Queries computables

Consideramos la Teoría de Modelos Finitos como marco teórico para estudiar las *consultas*, o *queries*, a bases de datos. Llamaremos *esquema de base de datos* a todo vocabulario relacional σ, e *instancia de base de datos* a toda σ-estructura. De esta manera, consideraremos como *base de datos* a toda clase numerable (finita o infinita) de estructuras relacionales finitas de un cierto vocabulario relacional finito.

Siguiendo a [CHANDRA 80], definiremos como *consulta* o *query* de aridad *r*, para algún entero $r > 0$, a toda función parcial *q* de la siguiente forma, donde σ es un vocabulario: $q : E_{\sigma, fin} \rightarrow E_{\langle R \rangle}$, donde la aridad de *R* es *r*, y tal que para toda σ-estructura *B*, $dom(q(B)) \subseteq dom(B)$.

Un *query booleano*, o *propiedad* es una función de la forma: $q : E_{\sigma, fin} \rightarrow \{\top, \bot\}$

Se dice que *q* es un *query computable*, si es una función recursiva parcial, en cualquier representación de las σ-estructuras en estructuras totalmente ordenadas, y si preserva isomorfismos.

Denotamos con *CQ* la clase de todos los queries computables. Y diremos que un lenguaje es *completo* si expresa, o computa, exactamente todos los queries en la clase *CQ*.

Si *C* es una clase de σ-estructuras, diremos que un lenguaje *se comporta como completo con C*, si computa todos los queries de la clase *CQ* restringida al vocabulario σ y a la clase *C*, es decir, si para todo query $q \in CQ$ de la forma $q: E_{\sigma, fin} \rightarrow E_{\langle R \rangle}$, donde *R* es de aridad *r* si *q* es de aridad *r*, para cualquier entero $r > 0$, computa el query $q' : E_{\sigma, fin\,|\,C} \rightarrow E_{\langle R \rangle}$. Y si para todo query $q \in CQ$ de la forma $q : E_{\sigma, fin} \rightarrow \{\top, \bot\}$, donde *q* es de aridad 0, computa el query $q' : E_{\sigma, fin\,|\,C} \rightarrow \{\top, \bot\}$. También diremos en este caso que la clase de queries definida de este modo es la *clase de queries computables sobre C*.

3. PRESENTACIÓN DE UNA APLICACIÓN

Tómese una base de datos que modelice redes de computadoras. En la misma se consideran los puntos de conexión, los cuales pueden ser de diferentes tipos de dispositivos como computadoras, distribuidores, puentes , etc., y sus conexiones directas. Entonces se tiene:

El vocabulario:

σ = < Nodos$^{(1)}$, Tipos $^{(1)}$, Tipo-Máquina$^{(1)}$, Conexión-directa $^{(2)}$, Tipo-por-nodo $^{(2)}$, Igual$^{(2)}$ >

La σ-estructura:

$\boldsymbol{B}$ = <D^B, Nodos^B, Tipos ^{B}, Tipo-Máquina^B, Conexión-directa ^{B}, Tipo-por-nodo ^{B}, Igual^B >

Dado el dominio de la estructura $\boldsymbol{D^B}$ = {nodo$_1$,..., nodo$_p$, Anfitrión, Servidor, Cliente, Colector, Distribuidor, Puente, Repetidor, Ruteador}†, con $\boldsymbol{N= |D^B| = p + 8,}$ con p arbitrario.

Dadas las siguientes instancias de las relaciones:

Nodos$^{\boldsymbol{B}}$ ={ $x \in D^B$ |"x es un punto de conexión"}={nodo$_1$, nodo$_2$,..., nodo$_p$}, donde | Nodos^B | = p.

Tipos$^{\boldsymbol{B}}$ **=** {Anfitrión, Servidor, Cliente, Colector, Distribuidor, Puente, Repetidor, Ruteador}, donde |Tipos^B | = 8.

Tipo-Máquina$^{\boldsymbol{B}}$ = {Anfitrión, Servidor, Cliente}, donde |Tipo-Máquina ^{B} | = 3.

Los elementos que pertenecen a Tipos y no pertenecen a Tipo-Máquina, son considerados *Dispositivos.*

Conexión-directa$^{\boldsymbol{B}}$ = {(x,y) | x,$y \in$ Nodos^B y "el nodo x tiene una conexión directa con el nodo y"}.

Dado que esta relación es simétrica, por ser la conexión bidireccional, trabajaremos con un generador minimal‡ de la misma. Además, asumiremos que los pares se arman de la siguiente forma: *Máquina-Máquina, Máquina-Dispositivo o Dispositivo-Dispositivo.*

Tipo-por-nodo$^{\mathbf{B}}$ = {(x,y) | $x \in$ Tipos^B y $y \in$ Nodos^B y "y es de tipo x" }

= {(tipo$_i$, nodo$_1$),(tipo$_j$, nodo$_2$),..., (tipo$_h$, nodo$_p$) } con $1 \leq i, j, h \leq 8$.

Esta relación es completa respecto de Nodos. Es decir, tomando la proyección de la relación en la segunda componente, obtenemos la relación Nodos^B.

Igual$^{\boldsymbol{B}}$={(x,x)|$x \in D^B$}={(nodo$_1$,nodo$_1$),...,(nodo$_p$,nodo$_p$),(Anfitrión,Anfitrión),..., (Ruteador, Ruteador)}, donde | Igual^B | = N.

Modelizaciones de topologías

Vamos a asumir, sin pérdida de generalidad, que se siguen las siguientes reglas en la modelización de la red deseada en la Base de Datos:

a) Caso 1:

Las máquinas M_1,...,M_q están conectadas entre sí mediante una topología Anillo. Entonces, en nuestra modelización, quedan los siguientes pares: (M_1, M_2), (M_2, M_3),...,(M_{q-1}, M_q), (M_q, M_1).

b) Caso 2:

Las máquinas M_1,...,M_q están conectadas entre sí mediante una topología Estrella a través del dispositivo *Di*. Entonces en nuestra modelización quedan los pares (M_1, D_i), (M_2, D_i),..., (M_{q-1}, D_i), (M_q, D_i).

---

† D^B = {node$_1$,..., node$_p$, Host, Server, WorkStation, Bus, Hub, Bridge, Repeater, Router}

‡ Un generador minimal de una relación simétrica es una relación, minimal en cantidad de tuplas, tal que a partir de ella se puede recuperar la relación original sabiendo que ella tenía la propiedad simétrica. Es decir, en el generador si se mantiene el par (x, y) no se mantendrá el par (y, x).

c) Caso 3:

Las máquinas $M_1,...,M_q$ están conectadas entre sí mediante una topología Colector a través del dipositivo colector C_i. Entonces, en la modelización quedan los pares (M_1, C_i), (M_2, C_i),..., (M_{q-1}, C_i), (M_q, C_i). Es semejante al caso 2, pero específicamente el dispositivo es colector, mientras que en el caso anterior está referido a diferentes tipos de dispositivo.

d) Caso 4:

Topologías heterogéneas se logran por composición de los criterios anteriores. Las conexiones entre subredes de diferentes topologías se pueden realizar mediante dispositivos o máquinas. También, se pueden considerar topologías compuestas como el caso de una *Rueda de Carro*[§], en cuya modelización quedan los pares (M_1, C_i), (M_2, C_i),...,(M_{q-1}, C_i),(M_q, C_i), (M_1, D_i), (M_2, D_i),..., (M_{q-1}, D_i), (M_q, D_i).

4. LÓGICAS DE CLAUSURA TRANSITIVA**

Sea R una relación binaria sobre un conjunto M, $R \subseteq M^2$. La *clausura transitiva*†† $CT(R)$ de R está definida por :

$$CT(R) := \{(a, b) \in M^2 \mid \text{existe } n>0 \text{ y } e_0,...,e_n \in M \text{ tal que } a = e_0,\ b = e_n \text{ y para todo } i < n,\ (e_i, e_{i+1}) \in R\}$$

Y la *clausura transitiva determinística* $CTD(R)$ está definida por

$$CTD(R) := \{(a, b) \in M^2 \mid \text{existe } n > 0 \text{ y } e_0,...,e_n \in M \text{ tal que } a = e_0,\ b = e_n \text{ y para todo } i < n,\ e_{i+1} \text{ es el único } e \text{ para el cual } (e_i, e) \in R\}$$

Las lógicas de *Clausura Transitiva* $FO(CT)$ y de *Clausura Transitiva Determinística* $FO(CTD)$ son obtenidas cerrando la lógica de primer orden (FO) bajo la clausura transitiva y la clausura transitiva determinística de relaciones definibles, respectivamente. Además, las lógicas de clausuras transitivas son cerradas bajo las operaciones usuales de primer orden. Así podemos construir combinaciones booleanas de fórmulas de CT, podemos anidar cuantificadores de CT, etc.

Utilizaremos la notación v_x para el vector de variables $(x_1,...,x_n)$ y la longitud de dicho vector es n. Por abuso de notación, hablaremos igualmente de vectores de términos. Para evitar confusión, cuando sea necesario, aclararemos el tipo de elementos del vector.

Las reglas sintácticas de generación de fórmulas en estas lógicas son esencialmente las mismas que para FO, con el agregado de la siguiente regla particular en cada caso:

Para $FO(CT)$ la regla es:

- $\dfrac{\varphi}{[CT\ v_x\ v_y\ \varphi]v_s\ v_t}$ Donde las variables que aparecen en v_x y v_y son distintas de a pares y donde los vectores v_x, v_y, v_s, y v_t son todos de la misma longitud, y los vectores v_s y v_t son vectores de términos.

Para $FO(CTD)$ la última regla es reemplazada por:

- $\dfrac{\varphi}{[CTD\ v_x\ v_y\ \varphi]v_s\ v_t}$ Con las mismas condiciones.

Definimos:

$$libres([CT\ v_x\ v_y\ \varphi]v_s\ v_t) := libres(v_s) \cup libres(v_t) \cup (libres(\varphi) \setminus \{v_x, v_y\})$$

y similarmente para $FO(CTD)$.

El significado de $[CT\ v_{x,}v_y\ \varphi(v_x, v_y, v_u)]v_s\ v_t$ es que $(v_s, v_t) \in CT(\{(v_x, v_y) \mid \varphi(v_x, v_y, v_u)\})$ y el significado de $[CTDv_{x,}v_y\ \varphi(v_x, v_y, v_u)]v_s\ v_t$ es que $(v_s, v_t) \in CTD(\{(v_x, v_y) \mid \varphi(v_x, v_y, v_u)\})$.

---

§ *Rueda de carro*: composición de topología estrella y anillo, involucrando los mismos nodos: ⊛

** Para las definiciones y notación nos basaremos en [EBBINGHAUS 95], y en [GRÄDEL 95]

†† Es sólo una clausura transitiva y no una reflexo-transitiva

Aquí $\{(v_x, v_y) \mid \varphi(v_x, v_y, v_u)\}$ es considerado como una relación binaria sobre el conjunto de longitud(v_x)-tuplas del universo.

Cabe aclarar, que las variables libres denotadas por v_u no participan en la clausura transitiva, aunque participan en la fórmula φ. El cuantificador liga únicamente las variables indicadas, es decir v_x y v_y [EBBINGHAUS 95].

En [EBBINGHAUS 95] se muestra que, en el caso de estructuras finitas, $FO(CTD) \subseteq FO(CT)$. Esto puede verse con claridad porque:

$[CTDv_{x,}v_y\ \varphi(v_x, v_y)](v_u, v_v)$ es equivalente a $[CTv_{x,}v_y\ \varphi(v_x, v_y) \wedge \forall v_z\ (\varphi(v_x, v_z) \rightarrow v_y = v_z)](v_u, v_v)$

Decimos que una lógica *L captura* una clase de complejidad *C* sobre una clase de estructuras finitas *O*, si las consultas sobre *O* que son expresables en *L* coinciden con las consultas sobre *O* en la clase de complejidad *C*.

Sobre la clase de todas las estructuras finitas ordenadas([GRÄDEL 95])

(i) $FO(CTD)$ *captura* LOGSPACE

(ii) $FO(CT)$ *captura* NLOGSPACE.

Resolución de Queries

1. ¿Existen pares de nodos de la red no conectados entre sí?

   $\varphi \equiv \exists x\ (\ \exists y\ (\neg[CTv,z\ (Conexión\text{-}directa(v,z) \vee Conexión\text{-}directa(z,v))](x,y)))$

2. ¿Cuáles son los pares de máquinas conectadas entre sí?

   En primera instancia obtengamos los nodos del dominio que son máquinas, mediante la siguiente fórmula:

   $\theta(x) \equiv \exists w\ (Tipo\text{-}por\text{-}nodo(w,x) \wedge Tipo\text{-}máquina(w))$

   Entonces, a continuación, obtenemos las conexiones máquina a máquina:

   $\varphi(x,y) \equiv [CTv,z((Conexión\text{-}directa(v,z) \vee Conexión\text{-}directa(z,v))](x,y) \wedge \theta(x) \wedge \theta(y)$

3. ¿Cuáles son los pares de máquina conectadas entre sí sólo por dispositivos?

   En primera instancia, obtengamos los nodos del dominio que son máquinas mediante la siguiente fórmula:

   $\theta(x) \equiv \exists w\ (Tipo\text{-}por\text{-}nodo(w,x) \wedge Tipo\text{-}máquina(w))$

   Luego, obtengamos los pares de dispositivos conectados entre sí sólo por dispositivos mediante la siguiente fórmula:

   $\gamma(x,y) \equiv [CTv,z((Conexión\text{-}directa(v,z) \vee Conexión\text{-}directa(z,v)) \wedge \neg\theta(v) \wedge \neg\theta(z)\ )](x,y)$

   Después, obtengamos todos los pares de nodos tales que la primera componente es una máquina y la segunda un dispositivo:

   $\psi(x,y) \equiv\ \theta(x) \wedge \neg\ \theta(y)$

   Y finalmente, resolvemos:

   $\varphi(x,y) \equiv \exists z\ (\ \exists v\ (Conexión\text{-}directa(x,z) \wedge Conexión\text{-}directa(y,v) \wedge \gamma(z,v) \wedge \psi(x,z) \wedge \psi(y,v)))$

4. ¿Cuáles son las conexiones críticas entre máquinas?

   Entendiéndose como conexiones críticas aquellos nodos conectados entre sí por una única vía. En primera instancia, obtengamos los nodos del dominio que son máquinas mediante la siguiente fórmula:

   $\theta(x) \equiv \exists w\ (Tipo\text{-}por\text{-}nodo(w,x) \wedge Tipo\text{-}máquina(w))$

   Entonces, la fórmula final es:

   $\varphi(x,y) \equiv [CTu,w([CTDv,z(Conexión\text{-}directa(v,z)](u,w) \vee$
   $[CTDv,z(Conexión\text{-}directa(v,z)](w,u)](x,y) \wedge \theta(x) \wedge \theta(y)$

5. ¿Existen subredes con topología de anillo?
   En primera instancia, obtengamos los nodos del dominio que son máquinas mediante la siguiente fórmula:
   $\theta(x) \equiv \exists w$ (Tipo-por-nodo(w,x) $\wedge$ Tipo-máquina(w))

   Entonces:
   $\varphi \equiv \exists x$ ([CTv,z ((Conexión-directa(v,z) $\vee$ Conexión-directa(z,v)) $\wedge$ $\theta(v) \wedge \theta(z)$)](x,x))

6. ¿Cuáles son las máquinas conectadas a alguna subred con topología de anillo?
   En primera instancia, obtengamos los nodos del dominio que son máquinas mediante la siguiente fórmula:
   $\theta(x) \equiv \exists w$ (Tipo-por-nodo(w,x) $\wedge$ Tipo-máquina(w))

   Finalmente:
   $\varphi(x) \equiv$[CTv,z ((Conexión-directa(v,z) $\vee$ Conexión-directa(z,v)) $\wedge$ $\theta(v) \wedge \theta(z)$)](x,x)

5. LÓGICAS DE PUNTO FIJO[‡‡]

Introducción

Sea $\varphi(v_x, S)$ una fórmula de primer orden en la cual v_x es una r-tupla de variables y S es un símbolo de relación r-aria (no incluida en un vocabulario σ) y sea B una estructura sobre el vocabulario σ. La fórmula φ da lugar a un operador $\Phi(S)$ desde relaciones r-arias sobre el universo D^B de B a relaciones r-arias sobre B, donde $\Phi(T) = \{v_a \mid B \vDash \varphi(v_a, T)\}$, para cada relación r-aria T sobre D^B.

Cada uno de tales operadores $\Phi(S)$ genera una secuencia de *etapas* que son obtenidas iterando $\Phi(S)$. Las etapas Φ^m (también denotadas por φ^m), con $m \geq 1$, de Φ sobre B, están definidas por inducción: $\Phi^1 = \Phi(\varnothing)$, $\Phi^{m+1} = \Phi(\Phi^m)$. Intuitivamente, uno gustaría asociar con un operador $\Phi(S)$ el "límite" de sus etapas. Esto es posible sólo cuando la secuencia Φ^m, con $m \geq 1$, de las etapas "converge", es decir, cuando existe un entero m_0 tal que $\Phi^{m_0} = \Phi^{m_0 + 1}$ y, de aquí, $\Phi^{m_0} = \Phi^m$, para todo $m \geq m_0$. Notar que, en este caso, Φ^{m_0} es un punto fijo de $\Phi(S)$, dado que $\Phi^{m_0} = \Phi^{m_0 + 1} = \Phi(\Phi^{m_0})$. La secuencia de etapas puede no ser convergente. En particular, esto ocurrirá si la fórmula $\varphi(v_x, S)$ no tiene punto fijo.

Lógica de Punto Fijo Inflacionaria

Una fórmula $\varphi(v_x, S)$ es *inflacionaria en S* si $T \subseteq \Phi(T)$ para cualquier relación r-aria T. En particular, φ es inflacionaria en S si es de la forma $S(v_x) \vee \psi(v_x, S)$. Si φ es inflacionaria en S, entonces la secuencia Φ^m, con $m \geq 1$, de etapas es incrementante. Así, debe tener un "límite".

Más precisamente, si B es una estructura finita con n elementos, entonces existe un entero $m_0 \leq n^r$ tal que $\Phi^{m_0} = \Phi^m$ para cada $m \geq m_0$. Es decir, la secuencia de etapas de $\varphi(v_x, S)$ converge a Φ^{m_0}. Escribimos φ^∞ o Φ^∞ para denotar el punto fijo Φ^{m_0} de φ. *La lógica de Punto Fijo Inflacionaria FO(PFI)* es la lógica de primer orden aumentada con la regla de formación de punto fijo inflacionario para fórmulas inflacionarias[§§].

[‡‡] Para las siguientes definiciones nos basamos en [ABITEBOUL 96]

[§§] De aquí en adelante, asumimos sin pérdida de generalidad, que cada fórmula inflacionaria es de la forma

$$S(x_1,..., x_n) \vee \psi(x_1,..., x_n)$$

Observación:

Se puede garantizar que el punto fijo también existe cuando la fórmula $\varphi(v_x, S)$ es *positiva* en *S*, esto es, cuando cada ocurrencia de *S* está gobernada por un número para de negaciones. En este caso, la secuencia Φ^m, con $m \geq 1$, de etapas es también incrementante, y consecuentemente tiene un límite que es un punto fijo. De hecho, ese límite es el menor punto fijo de φ. La *lógica de punto fijo positiva* es la lógica de primer orden aumentada con la regla de formación de menor punto fijo para fórmulas positivas. Es simple ver que la lógica *FO*(*PFI*) es tan expresiva como lo es la lógica de punto fijo positiva.

Es conocido que *FO*(*PFI*) captura la clase de complejidad PTIME.

Lógica de Punto Fijo No Inflacionaria

Podemos obtener lógicas con construcciones de iteración más expresivas que la lógica de punto fijo inflacionaria. Una lógica más poderosa resulta si se iteran operadores de primer orden generales, hasta alcanzar un punto fijo, el cual podría nunca ocurrir. En este caso podemos tener computaciones que nunca terminen.

Sea Φ^m, con $m \geq 1$, la secuencia de etapas del operador $\Phi(S)$ asociado con la fórmula $\varphi(x_1, ..., x_n, S)$. Si existe un entero m_0 tal que $\Phi^{m_0} = \Phi^{m_0+1}$, entonces ponemos $\varphi^\infty = \Phi^\infty = \Phi^{m_0}$; en otro caso, colocamos $\varphi^\infty = \Phi^\infty = \varnothing$.[***] Llamamos φ^∞ al *punto fijo no inflacionario* de φ sobre *B*. La *lógica de Punto Fijo No inflacionaria FO*(*PFN*) es la lógica de primer orden aumentada con la regla de formación de punto fijo no inflacionario para fórmulas de primer orden arbitrarias. Notar que la lógica de punto fijo *FO*(*PFN*) es una extensión de *FO*(*PFI*). Mientras que las fórmulas *FO*(*PFI*) pueden ser evaluadas en tiempo polinomial, *FO*(*PFN*) puede expresar problemas PSPACE-completos.

Se sigue que *FO*(*PFN*) captura la clase de complejidad PSPACE [VARDI 82].

Claramente, si *FO*(*PFI*) y *FO*(*PFN*) tuvieran la misma potencia expresiva, entonces P = PSPACE. No es claro, sin embargo, que la inversa se mantenga debido a que *FO*(*PFI*) y *FO*(*PFN*) necesitan orden para "capturar completamente" a P y a PSPACE respectivamente. Más aún, P = PSPACE si y sólo si *FO*(*PFI*) y *FO*(*PFN*) tienen la misma potencia expresiva [ABITEBOUL 91; ABITEBOUL 95a].

FO(*PFI*) y *FO*(*PFN*) son obtenidas iterando cuantificadores de primer orden inflacionarios y no inflacionarios, respectivamente. En ambos casos, sin embargo, la iteración es secuencial y determinística.

Observaciones

Las lógicas de punto fijo pueden ser parametrizadas de acuerdo a la potencia de sus cuantificadores de primer orden, *inflacionario* vs. *no inflacionario*. Usamos la notación *PF*α, para referirnos a una lógica de punto fijo con cuantificadores de tipo α (α ∈{*I*, *N*}). Así, las lógicas *FO*(*PFI*) y *FO*(*PFN*) se denotarán como *PFI* y *PFN* respectivamente.

Sea α como antes; las *fórmulas* en *PF*α son obtenidas partiendo desde los átomos, por aplicaciones repetidas de cuantificadores de primer orden (¬,∨,∧,→,∃,∀) y los cuantificadores de punto fijo. La regla de formación de punto fijo es la familiar, salvo que se agrega la siguiente regla en cada caso:

Para un vocabulario σ, en *PFI* considerar la siguiente regla:

- $\dfrac{\varphi}{[PFIv_{v,X}\ \varphi]\ v_t}$ donde las longitudes de v_x y v_t son las mismas y coinciden con la aridad de *X*, siendo v_t un vector de términos.

[***] Realmente, se muestra en [ABITEBOUL 90] que, no existe pérdida de generalidad en restringir la atención a cuantificadores de primer orden convergentes.

Para *PFN* la última regla es reemplazada por:

- $\dfrac{\varphi}{[PFN v_{x,X}\ \varphi]\ v_t}$ donde las longitudes de v_x y v_t son las mismas y coinciden con la aridad de X, siendo v_t un vector de términos.

Las *sentencias* son fórmulas sin variables libres de primer y segundo orden, donde la ocurrencia libre de variables está definida de la manera estándar, agregando, por ejemplo, para *PFI* la cláusula:

$$libres([PFIv_{x,X}\ \varphi]\ v_t) := libres(v_t) \cup (libres(\varphi) \setminus \{v_x, X\})$$

La semántica es definida inductivamente con respecto al calculo anterior, el significado de $[PFIv_{x,X}\ \varphi]v_t$ siendo que $v_t \in \varphi^{\infty}$,y que $[PFNv_{x,X}\ \varphi]v_t$ siendo, de manera similar, que $v_t \in \varphi^{\infty}$. Más precisamente: Si X es k-aria y si las variables libres en $[PFIv_{x,X}\ \varphi]\ v_t$ están entre v_u e Y, y v_b y S son interpretaciones en B de v_u e Y, respectivamente, entonces

$$B \vDash [PFIv_{x,X}\ \varphi]v_t\ [v_b, S] \quad \text{sii} \quad (t_1[v_b],..., t_k[v_b]) \in \varphi[S]^{\infty},$$
$$B \vDash [PFNv_{x,X}\ \varphi]v_t\ [v_b, S] \quad \text{sii} \quad (t_1[v_b],..., t_k[v_b]) \in \varphi[S]^{\infty},$$

Se pueden demostrar, para estas lógicas de punto fijo estándar, resultados tales como la clausura bajo composición y complemento. Se mantiene en ambas lógicas el colapso de los anidamientos de puntos fijos, es decir que es suficiente con un cuantificador de punto fijo por fórmula, y la inducción simultánea, o sea que es suficiente con una sola variable de relación. Estos resultados fueron demostrados para *PFI* en [GUREVICH 86] y para *PFN* en [ABITEBOUL 91a]. Las inclusiones entre estas lógicas son:

$$CTD \subseteq CT \subseteq PFI \subseteq PFN$$

Resolución de Queries

1. ¿Existen pares de nodos de la red no conectados entre sí?
   μ es un una expresión de *SO* de la siguiente forma:
   $\mu(x,y,X) \equiv$ *Conexión-directa(v,z)*$\vee$*Conexión-directa(z,v)*$\vee X(v,z) \vee$
   $\exists w(X(v,w) \wedge$(*Conexión-directa(w,z)*$\vee$*Conexión-directa(z,w))*

   La relación X irá variando su contenido en cada paso, hasta converger en su punto fijo. Entonces, la expresión final es:
   $\varphi \equiv \exists x\ (\ \exists y\ (\neg[PFIv,z,X\mu(v,z,X)](x,y)))$

2. ¿Cuáles son los pares de máquinas conectadas entre sí?
   En primera instancia calculamos los nodos del dominio que son máquinas:
   $\theta(x) \equiv \exists w$ *(Tipo-por-nodo(w,x)* $\wedge$ *Tipo-máquina(w))*

   Sea μ es un una expresión de *SO* de la siguiente forma:
   $\mu(x,y,X) \equiv$ *Conexión-directa(v,z)*$\vee$*Conexión-directa(z,v)*$\vee X(v,z) \vee$
   $\exists w(X(v,w) \wedge$(*Conexión-directa(w,z)*$\vee$*Conexión-directa(z,w))*

   Finalmente:
   $\varphi(x,y) \equiv [PFIv,z,X\mu(v,z,X)](x,y) \wedge \theta(x) \wedge \theta(y)$
   O bien
   $\varphi(x,y) \equiv [PFNv,z,X\mu(v,z,X) \vee X(v,z)](x,y) \wedge \theta(x) \wedge \theta(y)$

3. ¿Cuáles son los pares de máquina conectadas entre sí sólo por dispositivos?
   En primera instancia calculamos los nodos del dominio que son máquinas:
   $\theta(x) \equiv \exists w$ *(Tipo-por-nodo(w,x)* $\wedge$ *Tipo-máquina(w))*

γ es un query binario que obtiene los pares de dispositivos conectados entre sí sólo por dipositivos.

$\gamma(x,y) \equiv [PFIv,z,X\ (\mu(v,z,X) \wedge \neg\theta(v) \wedge \neg\theta(z))](x,y)$

Sea ψ es un query binario que obtiene los todos los pares de nodos tales que la primera componente es una máquina y la segunda un dispositivo.

$\psi(x,y) \equiv \theta(x) \wedge \neg\ \theta(y)$

Finalmente, la fórmula sería:

$\varphi(x,y) \equiv \exists z\ (\ \exists v\ (Conexión\text{-}directa(x,z) \wedge Conexión\text{-}directa(y,v) \wedge \gamma(z,v) \wedge \psi(x,z) \wedge \psi(y,v)))$

4. ¿Existen subredes con topología de anillo?

   En primera instancia calculamos los nodos del dominio que son máquinas:

   $\theta(x) \equiv \exists w\ (Tipo\text{-}por\text{-}nodo(w,x) \wedge Tipo\text{-}máquina(w))$

   Sea μ es un una expresión de *SO* de la siguiente forma:

   $\mu(x,y,X) \equiv Conexión\text{-}directa(v,z) \vee Conexión\text{-}directa(z,v) \vee \exists w(X(v,w) \wedge (Conexión\text{-}directa(w,z) \vee Conexión\text{-}directa(z,w))$

   Finalmente, la fórmula sería:

   $\varphi \equiv \exists x\ (\ [PFIv,z,X(\mu(v,z,X) \wedge \theta(v) \wedge \theta(z))](x,x)$

5. ¿Cuáles son las máquinas conectadas a alguna subred con topología de anillo?

   En primera instancia calculamos los nodos del dominio que son máquinas:

   $\theta(x) \equiv \exists w\ (Tipo\text{-}por\text{-}nodo(w,x) \wedge Tipo\text{-}máquina(w))$

   Sea μ es un una expresión de *SO* de la siguiente forma:

   $\mu(x,y,X) \equiv Conexión\text{-}directa(v,z) \vee Conexión\text{-}directa(z,v) \vee \exists w(X(v,w) \wedge (Conexión\text{-}directa(w,z) \vee Conexión\text{-}directa(z,w))$

   Entonces, la fórmula final es:

   $\varphi(x) \equiv [PFIv,z,X(\mu(v,z,X) \wedge \theta(v) \wedge \theta(z))](x,x)$

6. Con el siguiente query buscamos ir eliminando paso a paso los puntos fronteras y en cada vuelta restringiendo la red a los nodos restantes. Finalmente, los nodos resultantes al query son aquellos que en una topología estrella es el nodo central; en una topología anillo, quedan todos; y en una topología colector queda el dispositivo colector. Para las heterogéneas puede interpretarse como la obtención de un grupo de nodos que conforman un núcleo dentro de la red.

   Restricción de la red quitando las capas de puntos fronteras.

   $\lambda(x,y,X) \equiv ((Conexión\text{-}directa(v,z) \vee Conexión\text{-}directa(z,v)) \wedge \neg\ X\ (x) \wedge \neg\ X\ (y))$

   Con la siguiente expresión calculamos los puntos frontera de la red restringida a la relación *X:*

   $\beta(x,X) \equiv \exists z_1\ (\lambda(x, z_1, X) \wedge \neg\exists z_2(\ \lambda(\ z_2, x, X) \wedge \neg\exists z_3(\ \lambda(\ x, z_3, X) \wedge \neg\ Igual(z_3, z_1)))) \vee$
   $\exists z_1\ (\lambda(z_1, x, X) \wedge \neg\exists z_2(\ \lambda(\ x, z_2, X) \wedge \neg\exists z_3(\ \lambda(\ z_3, x,\ X) \wedge \neg\ Igual(z_3, z_1))))$

   Y finalmente:

   $\varphi(x) \equiv (\ Nodos(x) \wedge \neg\ (\ [PFNv,X\ (\beta(v, X) \vee X\ (v))](x)))$

CONCLUSIONES

Dado que el poder expresivo de una Lógica de Primer Orden está restringido a una subclase de la clase de Queries Computables (*QC*), hemos utilizado una lógica extendida con cuantificadores de Clausura Transitiva y Punto Fijo para la expresión y evaluación de queries que en la aplicación elegida resultan naturales de hacer, y que no son expresables en *FO*.

Así podemos hacer analogías con otras aplicaciones semejantes, en donde los lenguajes de consulta disponibles a las bases de datos resultan pobres en su expresividad con respecto a la clase CQ, y que por muy decorosos que se presenten, frente a queries del tenor presentados en este artículo no es posible su expresión, dado que no cuentan con la capacidad de los cuantificadores vistos.

Esto es relevante de destacar porque considerando la clase CQ, en general los lenguajes de consultas que se utilizan en la práctica no toman en cuenta el concepto de computabilidad y son incompletos respecto del mismo, o calculan queries no computables en su defecto. Obsérvese que estos problemas de expresividad y computabilidad no son sólo de interés teórico, sino que los mismos pueden causar serios errores en aplicaciones de cierta complejidad [TURULL 96].

En virtud de lo expuesto, la formulación de query de [CHANDRA 80] es una abstracción precisa y rigurosa del tipo de consulta que un programa de computadora debería computar. Nosotros hemos mostrado una parte de estos resultados mostrando simplemente como con FO no es suficiente para la expresión de queries de cierta categoría. Conocemos que son problemas característicos de grafos, conocemos los conceptos teóricos y conocemos las limitaciones, pero realmente, al momento de la aplicación ¿ recordamos unificarlos ?

En nuestro trabajo [MALDOCENA 99] continuamos con esta línea de desarrollo, en la idea de seguir agregando cuantificadores de mayor potencia como Punto Fijo No Determinístico y Alternado en sus versiones Inflacionaria y No Inflacionaria, buscando aplicaciones de interés y analizando las limitaciones relevantes. Así podemos observar que sobre estructuras finitas ordenadas, con las extensiones de tales cuantificadores, capturaríamos la clase EXPTIME [ABITEBOUL 96], mientras que con las extensiones presentadas en el presente artículo sólo capturamos PSPACE [VARDI 82].

REFERENCIAS BIBLIOGRÁFICAS

[ABITEBOUL 90] Abiteboul, S. y Vianu, V.; "*Non-deterministic Languages to express Deterministic Transformation*". Proc. 9th ACM Symp. on Principles of Databases Systems, 218-229, 1990.

[ABITEBOUL 91] Abiteboul, S. y Vianu, V.; "*Generic Computation and its Complexity*", In Proc. 23rd STOC, 209-219, 1991.

[ABITEBOUL 95a] Abiteboul, S. y Vianu, V.; "*Computing with First Order Logic*", Journal of Computer and System Sciences, 50:309-335, 1995;

[ABITEBOUL 95b] Abiteboul,S; Hull, R. y Vianu, V.; "*Foundations of Databases*". Addison-Wesley Publishing Company, 1995.

[ABITEBOUL 96] Abiteboul, S.; Vardi, M. y Vianu, V.; "*Fixpoint Logics, Relational Machines,and Computational Complexity*", Versión Completa obtenida en Internet,1996

[CHANDRA 80] Chandra, A.K. y Harel, D.; "*Computable Queries for Relational Data Bases*". Journal of Computer and System Sciences 21, 156-178. 1980.

[EBBINGHAUS 84] Ebbinghaus, H.; Flum, J. y Thomas, W.; "*Mathematical Logic*", Springer-Verlag, 1984.

[EBBINGHAUS 95] Ebbinghaus, H.y Flum, J.; "*Finite Model Theory*", Springer-Verlag, 1995.

[GRÄDEL 95] Grädel, E. y McColm, G. L.; "*On the Power of Deterministic Closures"* Versión Final, Information and Computation, Vol. 119, 129-135, 1995

[GUREVICH 86] Gurevich, S.y Shelah, S.; "*Fixed Point Extensions of First Order Logic"* Annals of Pure and Applied Logic; 32:265-280, 1986.

[MALDOCENA 99] Maldocena, P. y Reyes, N., "*Expresiones de consultas a bases de datos mediante extensiones de lógica de primer orden*" Informe de Tesis de Licenciatura en Cs. de la Computación, U.N.S.L., 1999.

[TURULL 96] Turull Torres, J. M.; "*Clases de Bases de Datos L-Rígidas y Expresividad de Lenguajes Relacionales Incompletos*". Tesis Doctoral, UNSL, 1996.

[ULLMAN 88] Ullman, J. D.; "*Principles of Database and Knowledge Base Systems*". Computers Science Press, 1988.

[VARDI 82] Vardi, M. Y.; "*The Complexity of relational queries languages*"Proc. 14th ACM Symp. on Theory of Computing, 137-146, 1982.