Un Intérprete de Consultas a Bases de Datos Expresadas Mediante Lógica de Primer Orden con Clausura Transitiva

Nora Reyes, Alejandro Grosso, Paulino Maldocena, J.M. Turull Torres [1]

UNIVERSIDAD NACIONAL DE SAN LUIS
Departamento de Informática
Ejército de los Andes 950- Locales 107 y 109
5700 - San Luis
Argentina
Fax: 02652-430224

{nreyes, agrosso, pmaldo, turull} @unsl.edu.ar

Resumen

Este trabajo consiste en la descripción de un intérprete de consultas a Bases de Datos relacionales expresadas utilizando fórmulas de la lógica de Primer Orden (*FO*) extendida con cuantificadores de clausura transitiva (*CT*) y clausura transitiva determinística (*CTD*), que suponen iteraciones en su interpretación semántica, que están ausentes en FO.

La motivación es aumentar el grado de expresividad del lenguaje de consulta. En [CH80] se demostró que la expresividad de FO está estrictamente contenida en (y es muy inferior a) LOGSPACE. Al extender FO, con dichos cuantificadores de clausura, se capturan las clases de complejidad que se mencionan, sobre *estructuras finitas ordenadas* [GMc95]:

FO(CTD) *captura* LOGSPACE
FO(CT) *captura* NLOGSPACE

Así, al trabajar con estas extensiones de FO, se pueden analizar consultas a Bases de Datos que están en la clase NLOGSPACE.

El intérprete permite la evaluación de consultas expresadas con las extensiones de FO planteadas, dentro de un ambiente que además permite operar sobre la Base de Datos.

Palabras Claves: Lenguajes de Consulta a Bases de Datos, Lógica de Primer Orden, Clausura Transitiva, Bases de Datos.

[1] Universidad Tecnológica Nacional (FRBA); Universidad Nacional de San Luis; Subsidio de Universidad CAECE y Universidad Nacional de Luján. Director del Proyecto de Investigación sobre Teoría de la Computación de la UNSL.

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 los lenguajes que se consideran en general son 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)* [Ull88][AV95]. En este trabajo utilizaremos *FO* para la expresión de consultas a las Bases de Datos, dado que su poder expresivo es equivalente a AR y CR.

También presentaremos la definición de la clase de consultas computables, llamada *Computable Queries* (*CQ*) [CH80]. Para ello observamos el *Modelo Relacional* en el marco de la *Teoría de Modelos Finitos* [EF95], 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 [CH80] 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.

Utilizamos un lenguaje de consultas a Bases de Datos Relacionales que computa queries expresables en *FO* extendido con los *cuantificadores Clausura Transitiva* y *Clausura Transitiva Determinística*. Este lenguaje es un primer subconjunto del que se está desarrollando e implementado por nuestro grupo de trabajo, utilizándolo como lenguaje modelo para la expresión y evaluación de consultas a Bases de Datos Relacionales.

Mostramos algunos ejemplos de aplicación y analizamos queries que no pueden ser expresados en *FO* y que sí son expresables con las extensiones de *FO* planteadas.

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 semántica habitual 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 [EFT84].

La *relación de satisfacción* ($\models$) 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 $1 \leq s_j \leq n$, $1 \leq j \leq r$: $B \models \varphi(x_1,\ldots,x_r)[\, d_{s_1},...,d_{s_r}\,]$.

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 [CH80], 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: $q : E_{\sigma, fin} \rightarrow \{\bot,\top\}$

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' que es la restricción de q a la clase C (o sea, q': $E_{\sigma, fin \mid C} \rightarrow E_{\langle R \rangle}$). Y si para todo query $q \in CQ$ de la forma $q : E_{\sigma, fin} \rightarrow \{\bot, \top\}$, donde q es de aridad 0, computa el query q': $E_{\sigma, fin \mid C} \rightarrow \{\bot, \top\}$. También diremos en este caso que la clase de queries definida de este modo es la *clase de queries computables sobre C*.

Lógicas de clausura transitiva

Para las definiciones y notación nos basaremos en [EF95] y en [GMc95].

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

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

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

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

Las lógicas de Primer Orden con *Clausura Transitiva FO*(*CT*) y de Primer Orden con *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 $(x_1, \ldots, x_n)$ y donde que 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.

Para FO(CT) la reglas son:

Para un vocabulario σ la clase de fórmulas de FO(CT) de vocabulario σ está dada por el cálculo (usamos la notación abreviada

$$\frac{\varphi_1,...,\varphi_n}{\varphi}$$

para la cláusula "Si $\varphi_1,..., \varphi_n$ son fórmulas entonces φ es una fórmula)

- $\dfrac{}{\varphi}$ Donde φ es una fórmula atómica de primer orden sobre σ

- $\dfrac{\varphi}{\neg \varphi}$; $\dfrac{\varphi, \psi}{\varphi \wedge \psi}$; $\dfrac{\varphi, \psi}{\varphi \vee \psi}$; $\dfrac{\varphi, \psi}{\varphi \rightarrow \psi}$; $\dfrac{\varphi}{\exists x\, \varphi}$; $\dfrac{\varphi}{\forall x\, \varphi}$

- $\dfrac{\varphi}{[CTv_{x,}\, v_y\, \varphi]v_s\, v_t}$ Si $v_x = (x_1,...,x_r)$ e $v_y = (y_1,...,y_r)$ para todo i, $1 \leq i \leq r$, es $x_i \neq y_i$; y donde las tuplas v_x, v_y, v_s, y v_t son todas de la misma longitud, v_s y v_t siendo tuplas de términos[2].

[2] Como el vocabulario considerado es relacional y no existen ni símbolos de función ni símbolos de relación 0-arios (constantes), los únicos términos posibles son los formados sólo por una variable de individuo.

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

$$\frac{\varphi}{[CTDv_{x,}\, v_y\, \varphi]v_s\, v_t}$$

con las mismas condiciones.

Definimos las *variables libres* de una fórmula $[CTv_{x,}\, v_y\, \varphi]v_s\, v_t$ como :

$$libres([\text{CT}\ v_{x,}v_y\ \varphi]v_s\, v_t) := libres(v_s) \cup libres(v_t) \cup (libres(\varphi) - \{v_x, v_y\})$$

y, similarmente, para FO(CTD).

El significado de $[\text{CT}\ v_{x,}\, v_y\, \varphi(v_x, v_y, v_u)]v_s\, v_t$ es $(v_s, v_t) \in \text{CT}(\{(v_x, v_y)$ tal que $\varphi(v_x, v_y, v_u)\})$ y el significado de $[\text{CTD}\ v_{x,}\, v_y\, \varphi(v_x, v_y, v_u)]v_s\, v_t$ es $(v_s, v_t) \in \text{CTD}(\{(v_x, v_y)$ tal que $\varphi(v_x, v_y, v_u)\})$ (aquí $\{(v_x, v_y)$ tal que $\varphi(v_x, v_y, v_u)\}$ es considerada como una relación binaria sobre el conjunto de $longitud(v_x)$-tuplas del dominio, o sea sobre $D^{longitud(v_x)}$).

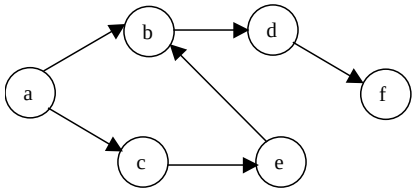
En [EF95] se muestra que, en el caso de estructuras finitas, FO(CTD) $\subseteq$ FO(CT) (Proposición 6.1.5 (a)). Esto puede verse claramente, porque la fórmula:
$[\text{CT}\ v_x, v_y\ \varphi(v_x, v_y) \wedge \forall v_z(\varphi(v_x, v_z) \rightarrow v_y = v_z)](v_u, v_v)$ equivale a la fórmula $[\text{CTD}\ v_{x,}v_y\ \varphi(v_x, v_y)](v_u, v_v)$.

Ejemplo: Un grafo puede verse como una estructura $G = \langle V, E \rangle$, donde E es una relación binaria sobre el conjunto de vértices V, o sea, $E \subseteq V^2$.

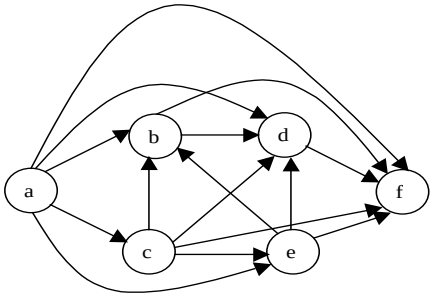
La expresión $[\text{CT}_{x,y}(E(x,y))](u,v)$ obtiene la clausura transitiva de la relación E.

La expresión $[\text{CTD}_{x,y}(E(x,y))](u,v)$ obtiene la clausura transitiva determinística de la relación E.

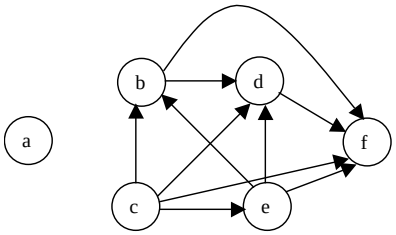
Sea el siguiente grafo:



La relación resultante, para el caso de clausura transitiva, gráficamente sería:



La relación resultante, para el caso de clausura transitiva determinística, gráficamente sería:



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

Se conocen caracterizaciones lógicas para las principales clases de complejidad. Para la clase PTIME y algunas otras clases de complejidad las caracterizaciones sólo se mantienen sobre la clase de *estructuras ordenadas finitas*, es decir, las estructuras que contienen entre sus relaciones un orden total "<" del dominio de la estructura (y/o una relación *sucesor*). Por ejemplo:

Sobre la clase de todas las estructuras finitas ordenadas

FO(CTD) *captura* LOGSPACE y FO(CT) *captura* NLOGSPACE [Imm87]

Es bien conocido que esto no se cumple sobre estructuras no ordenadas. Por ejemplo, en ese caso, FO(*CT*) no es capaz de expresar: "la cardinalidad del dominio de la estructura es par", una consulta que es LOGSPACE. De hecho esta consulta no es expresable aún en formalismos mucho más potentes, como las Máquinas Relacionales [AV91].

Descripción básica del Intérprete de Consultas

El intérprete de consultas a bases de datos relacionales, expresadas en lógica de primer orden extendida con los cuantificadores de Clausura Transitiva, consiste de dos etapas principales:

- El traductor de la entrada de la Base de Datos
- El evaluador de consultas.

Primero se debe traducir la entrada de una instancia de Base de Datos a archivos que contengan la información necesaria para una posterior evaluación de consultas de una manera eficiente. La etapa posterior es el desarrollo de un analizador sintáctico y semántico de consultas y el evaluador.

Para procesar la entrada de la instancia de la base de datos se hace una evaluación sintáctica y semántica de la misma; luego, si ésta está correctamente expresada, se procede a crear los archivos con la información necesaria para el intérprete de consultas.

Luego se usa el intérprete de consultas propiamente dicho. Dentro del mismo se pueden definir y evaluar consultas o definir macro-consultas. Para evaluar las consultas a la base de datos, primero se deben verificar sintácticamente (de acuerdo a la gramática establecida) y semánticamente; si fueron correctamente expresadas, se evalúan. La respuesta, o salida, será:

- en caso de ser un query *r*-ario: una relación de aridad *r* consistiendo de todas aquellas *r*-tuplas de elementos del dominio que hacen verdadero el query, ó
- en caso de query 0-ario: la respuesta podrá ser *Verdadero* o *Falso*.

El ambiente de trabajo permite ingresar las relaciones que conforman la Base de Datos, e ingresar y evaluar consultas. Todos los programas del Intérprete se han desarrollado en lenguaje C, y todo lo respectivo al ambiente se ha desarrollado utilizando DELPHI.

Describimos el diseño general del traductor de la Base de Datos y del intérprete de consultas y, más detalladamente, se describen los algoritmos utilizados por el evaluador de consultas para los cuantificadores con los que hemos extendido a FO.

TRADUCTOR DE LA BASE DE DATOS

En lo que sigue sea $B = \langle D^B, R_1^B, ..., R_s^B \rangle$ una Base de Datos Relacional, teniendo cada R_i aridad r_i, y siendo el dominio $D^B = \{d_1, d_2, ..., d_n\}$, y por lo tanto $|D^B| = n$.

La entrada a esta etapa es un texto que contiene la descripción de una Base de Datos, o sea, la información necesaria para obtener su esquema y su instancia.

En todas las Bases de Datos aparecerá automáticamente la relación *IGUAL* de aridad 2. Como su nombre lo indica, contendrá los pares de la forma (d_i,d_i) para $1 \leq i \leq n$, donde cada $d_i \in D^B$.

La Figura 1 sintetiza las entradas y salidas de la primera etapa: El *traductor de la Base de Datos* y muestra gráficamente como se procesa la información contenida en la descripción de la Base de Datos, para crear las estructuras necesarias para llevar a cabo la evaluación de consultas. Es decir, obtener esquema, dominio e instancia de la Base en estructuras de almacenamiento adecuadas para responder eficientemente las consultas.

El traductor realiza el análisis sintáctico (o parsing)[3] y el análisis semántico de la Base, y una vez que se ha verificado que la entrada es correcta, ésta se traducirá en las estructuras necesarias para esquema, dominio e instancia de la Base.

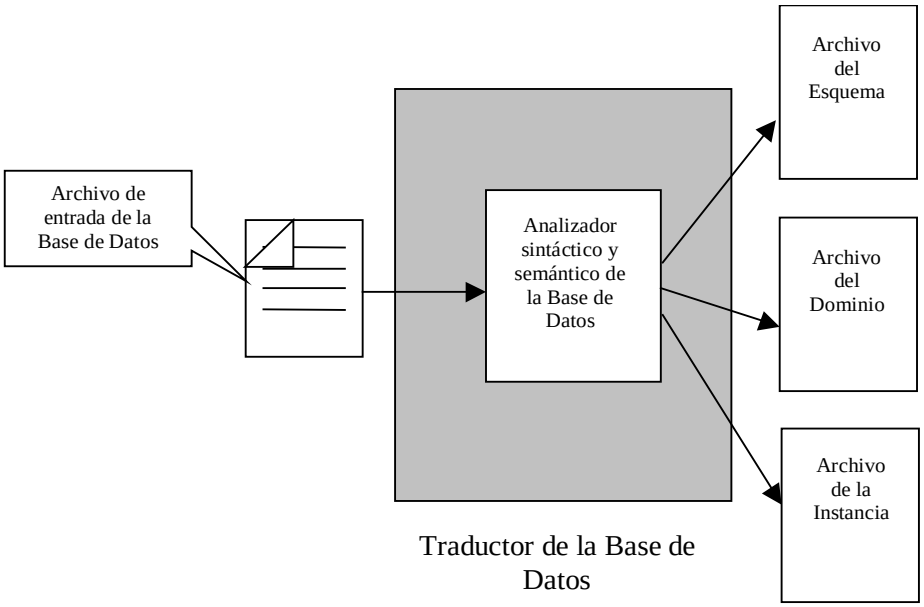


Figura 1: Esquema básico del Traductor

INTÉRPRETE DE CONSULTAS A LA BASE DE DATOS

La Figura 2 grafica las entradas y salidas de la segunda etapa, que es la interpretación de las consultas a la Base de Datos ingresada previamente.

En ella aparecen, además del archivo que contiene el texto de la consulta que se desea realizar, el archivo que contiene las consultas definidas como *macro-consultas* (*macros*) a la Base de Datos actual y los archivos de información de la Base de Datos. Como salida se muestra por pantalla el resultado de la consulta.

[3] El analizador sintáctico está realizado con el método descedente recursivo, y en particular predictivo [ASU86] y de acuerdo a la gramática para el ingreso de la base de datos.

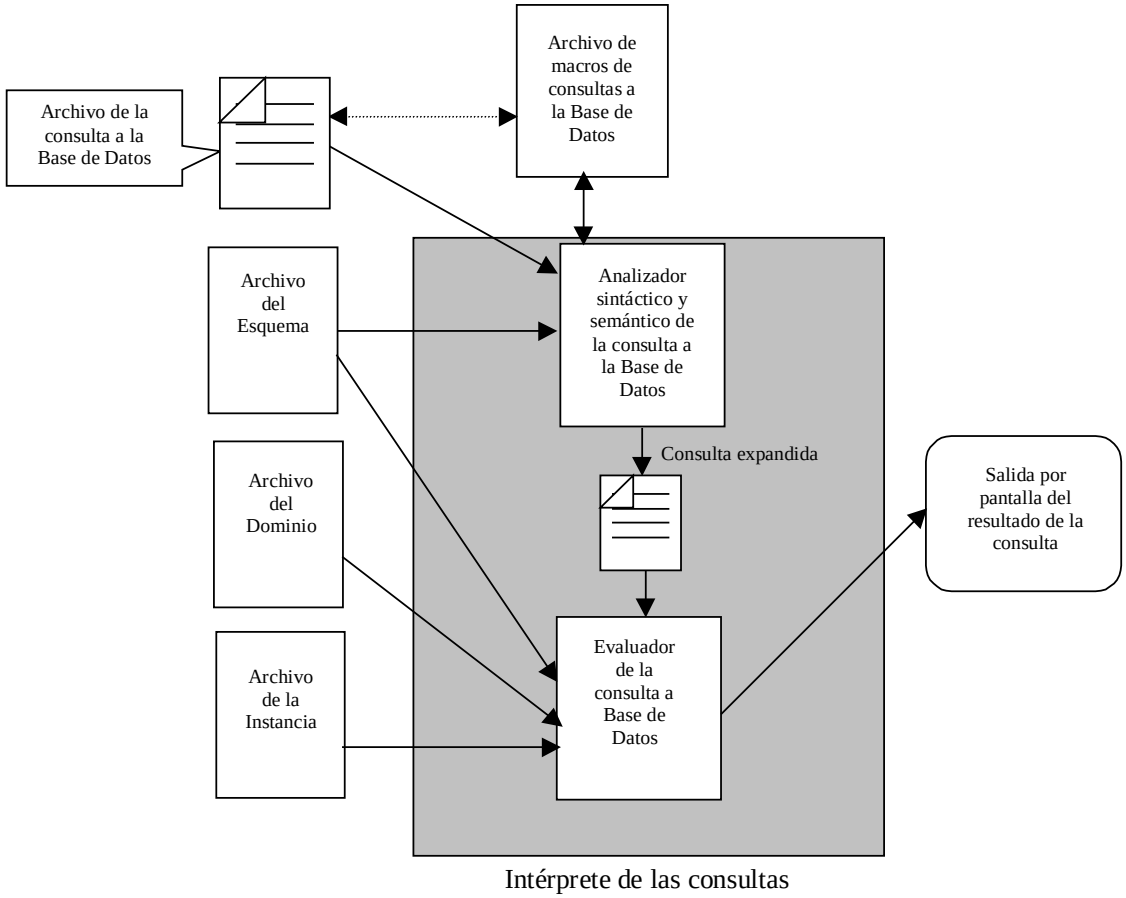


Figura 2: Esquema básico del Intérprete de Consultas

Una *macro-consulta* es una consulta, a la que se le asigna un nombre, permitiendo reusarla dentro de otras consultas mediante una invocación a la misma. Ésta no se evalúa al momento de definirla.

En caso de que la consulta ingresada esté correctamente expresada, la salida por pantalla mostrará una relación, de la aridad de la consulta planteada, o un valor verdadero o falso si la consulta era booleana, o sea, si la consulta era 0-aria.

En el intérprete pueden verse dos componentes importantes:

- Analizador sintáctico y semántico de la consulta
- Evaluador de la consulta

Una vez ingresada la consulta se realiza el análisis sintáctico y semántico de la misma, teniendo en cuenta la gramática para el lenguaje de consulta. Si ella es sintáctica y semánticamente correcta se evalúa (salvo en el caso que dicha consulta sea una macro); si no, se muestran los mensajes de error correspondientes.

Para realizar el análisis semántico de las consultas hacemos uso de la información de la Base de Datos generada por el traductor y de las macros incorporadas hasta el momento. El analizador sintáctico (o parser) es predictivo descendente recursivo y basado en la gramática del lenguaje de consulta soportado.

Para indicar que se está en presencia de la definición de una macro-consulta la consulta viene precedida por la palabra clave *DEF* .

Por cada variable ligada utilizada en la macro se mantiene la lista de posiciones en el texto en donde aparece; dado que, cuando ella sea invocada pueden ocurrir conflictos de nombres de variables entre la fórmula (o subfórmula) que invoca y la macro.

Este caso se presenta cuando la fórmula (o subfórmula) invocante tiene una variable libre que aparece ligada en el texto de la macro. Si no se sustituyera el nombre de la variable ligada, que se encuentra en esta situación, puede llevar a errores de interpretación. Obsérvese el siguiente ejemplo de macro-consultas y consulta[4], para ilustrar la situación planteada:

Macro:

DEF _F1(x1,x2) = ∀ x3(R2(x1,x3) ∨ R2(x2,x3)) ∨ R2(x1,x2)

DEF _F2(x3,x4,x2) = ∀ x5(R2(x3,x5) ∨ _F1(x4,x2)) ∨ R3(x3,x4,x2)

DEF _F3(x4) = ∀ x5(R2(x5,x4) ∧ _F2(x5,x4,x4)) ∨ ∀ x6(_F1(x4,x6)) ∧ R1(x4)

Consulta:

_F4(x1,x2) = R2(x1,x2) O PT x4(EX x5 (_F3(x5) O _F1(x4,x5) Y _F2(x4,x5,x1)))

Por ejemplo aquí se puede ver que la variable x5, que en la consulta aparece como libre, se utiliza como parámetro actual de la macro-consulta _F3, en donde aparece ligada por un cuantificador universal. Si se sustituyeran directamente los parámetros formales por los actuales se obtendría la siguiente fórmula:

∀ x5(R2(x5,x5) ∧ _F2(x5,x5,x5)) ∨ ∀ x6(_F1(x5,x6)) ∧ R1(x5)

Claramente ha cambiado el significado respecto de la fórmula de la macro. Para mantener la semántica correcta debe hacerse la sustitución de manera "inteligente"; o sea, teniendo en cuenta qué rol jugaba cada variable dentro de la macro-consulta, una variable que era ligada debe seguir siéndolo y una que era libre también. Por lo tanto, una adecuada sustitución sería por ejemplo:

PT x61(R2(x61,x5) Y _F2(x61,x5,x5)) O PT x6(_F1(x5,x6)) Y R1(x5)[5]

Así, se cambia el nombre de la variable ligada utilizada por la macro en todas las posiciones en donde aparece, utilizando para ello un nombre de variable no usado hasta el momento.

Dado que en el ejemplo hay invocaciones de macro-consultas anidadas, luego de todas las expansiones de macros, la expresión resultante a ser evaluada sería:

_F4(x1,x2) = R2(x1,x2) ∨ ∀x4(∃x5(∀x611(R2(x611,x5) ∧ ∀x61(R2(x611,x61) ∨

∀x3(R2(x5,x3) ∨ R2(x5,x3)) ∨ R2(x5,x5)) ∨ R3(x611,x5,x5)) ∨

∀x6(∀x3(R2(x5,x3) ∨ R2(x6,x3)) ∨ R2(x5,x6)) ∧ R1(x5) ∨

∀x3(R2(x4,x3) ∨ R2(x5,x3)) ∨ R2(x4,x5) ∧ ∀x6111(R2(x4,x6111) ∨

∀x3(R2(x5,x3) ∨ R2(x1,x3)) ∨ R2(x5,x1)) ∨ R3(x4,x5,x1)))

Algoritmos del evaluador

A continuación describiremos los algoritmos diseñados para evaluar las consultas expresadas en la extensión de FO considerada en este trabajo. Dichos algoritmos se invocan desde el evaluador de consultas el que, además, contiene lo necesario para evaluar FO.

Para cada uno de los cuantificadores que extienden a FO hay un algoritmo que realiza lo necesario para evaluarlo. A su vez, cada uno de ellos puede invocar recursivamente al evaluador.

---

[4] Suponiendo que todas las relaciones mencionadas pertenecen al esquema de la Base de Datos consultada.

[5] Sólo mirando _FA y _F3, x_{61} es un nombre de variable no utilizada.

Para la implementación de los algoritmos que evalúan los nuevos cuantificadores, los que se agregan a los ya conocidos de la lógica de primer orden, se tuvo en cuenta la definición formal del cuantificador y que estos algoritmos serán invocados durante la evaluación de una fórmula.

En todos los casos, el resultado de la aplicación de un cuantificador es una relación definida sobre el dominio de la estructura y cuya aridad depende del contexto de la fórmula.

Ahora veremos la descripción del algoritmo que evalúa cada cuantificador.

CUANTIFICADOR DE CLAUSURA TRANSITIVA

Al evaluar una fórmula φ, con $m+j$ variables libres, se obtiene como resultado el conjunto de $m+j$-tuplas de elementos del dominio que hacen verdadera a φ. Si el cuantificador liga m variables: x_1, ...x_m, y φ tiene $m+j$ variables libres: $x_1,...,x_{m+j}$, la relación resultante, a la que se le aplica el cuantificador, es la relación m-aria inducida por φ sobre el dominio D considerado[6]. Como m, en este caso, debe ser un número par, estas m-tuplas pueden tomarse como pares de k-tuplas que pertenecen a una relación binaria definida sobre D^k, siendo $k = m/2$.

El cuantificador de clausura transitiva permite obtener la clausura transitiva de la relación *binaria* inducida por la fórmula que actúa como parámetro, sobre k-tuplas del dominio de la base de datos (D^k). El algoritmo que usamos es una adaptación del ya conocido algoritmo de Warshall [PFA77] para cálculo de Clausura Transitiva de una relación binaria sobre un dominio, sólo que trabaja sobre k-tuplas.

CUANTIFICADOR DE CLAUSURA TRANSITIVA DETERMINÍSTICA

En este caso también, si el cuantificador se aplica a una fórmula φ, con $m+j$ variables libres, y si él liga m variables; la relación resultante, a la que se le aplica el cuantificador, es la relación m-aria inducida por φ sobre el dominio D considerado. Como m, en este caso, debe ser un número par, estas m-tuplas pueden tomarse como pares de k-tuplas que pertenecen a una relación binaria definida sobre D^k, siendo $k = m/2$.

El cuantificador de CTD permite obtener la clausura transitiva determinística de la relación *binaria* definida sobre k-tuplas del dominio, de la base de datos, por la fórmula que actúa como parámetro. El algoritmo también es una adaptación del algoritmo de Warshall para calcular la Clausura Transitiva de una relación binaria sobre un dominio, pero tiene en cuenta la restricción impuesta por el determinismo y también trabaja sobre k-tuplas de elementos del dominio.

La restricción para lograr determinismo se consigue considerando sólo los pares (x, y), de la relación definida por la fórmula, que cumplan que para x (como primer componente del par) y es el único elemento que lo acompaña en la relación. O sea, que se restringe la relación original definida por la fórmula a los pares que son determinísticos.

Luego, se calcula la clausura transitiva de la manera en que se hace con el cuantificador CT, pero esta vez sobre la restricción determinística de la relación.

Si llamamos R a la relación binaria inducida por la fórmula φ(v_x ,v_y ,v_u), donde v_x y v_y son k-tuplas y v_u una j-tupla de elementos del dominio, y R_D a la restricción determinística de R; se puede ver claramente que: CT(R_D) = CTD(R).

Ejemplo de 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 [GMR99]. Entonces se tiene:

El vocabulario:

$\sigma = \langle \text{Nodos}^{(1)}, \text{Tipos}^{(1)}, \text{Tipo-Máquina}^{(1)}, \text{Conexión-directa}^{(2)}, \text{Tipo-por-nodo}^{(2)}, \text{Igual}^{(2)} \rangle$

[6] Dicha relación es la proyección a $x_1,...,x_m$ de la relación $m+j$-aria inducida por φ sobre D.

La σ-estructura:

$B = \langle D^B, \text{Nodos}^B, \text{Tipos}^B, \text{Tipo-Máquina}^B, \text{Conexión-directa}^B, \text{Tipo-por-nodo}^B, \text{Igual}^B \rangle$

Dado el dominio de la estructura D^B = {nodo_1,..., nodo_p, Anfitrión, Servidor, Cliente, Colector, Distribuidor, Puente, Repetidor, Ruteador}[7], con $N = |D^B| = p + 8$, con p arbitrario.

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

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

2. ¿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:

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

   Entonces, la fórmula final es:

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

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 para la expresión y evaluación de queries que en diversas aplicaciones pueden resultar naturales de hacer, y que no son expresables en *FO*.

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 [Tur96].

En virtud de lo expuesto, la formulación de query de [CH80] 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.

En nuestro trabajo continuamos con esta línea de desarrollo, en la idea de seguir agregando cuantificadores de mayor potencia expresiva como distintas versiones de cuantificadores de punto fijo, 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 [AVV97], mientras que con las extensiones presentadas en el presente artículo sólo capturamos NLOGSPACE [GMc95].

[7] D^B = {node_1,..., node_p, Host, Server, WorkStation, Bus, Hub, Bridge, Repeater, Router}

Referencias Bibliográficas

[ASU86] Aho, Alfred; Ravi, Sethi; Ullman, Jeffrey; "*Compilers, Principles, Techniques, and Tools";* Addison-Wesley, 1986

[AV91] Abiteboul,S; Vianu, V.; "*Generic Computation and Its Complexity*", STOC 1991.

[AV95] Abiteboul, S.; Vianu, V.; "*Computing with First Orde r Logic*"; Journal of Computer and System Sciences, 50:309-335, 1995.

[AVV97] Abiteboul, S.; Vardi, M. y Vianu, V.; "*Fixpoint Logics, Relational Machines, and Computational Complexity*", Journal of ACM, 44(1), 30-56.Enero, 1997.

[CH80] Chandra, A.K.; Harel, D.; "*Computable Queries for Relational Data Bases*". Journal of Computer and System Sciences 21, 156-178. 1980.

[EF95] Ebbinghaus, H; Flum, J.; "*Finite Model Theory*", Springer-Verlag, 1995.

[EFT84] Ebbinghaus, H; Flum, J.;Thomas, W.; "*Mathematical Logic*", Springer-Verlag, 1984.

[GMc95] Grädel, E. y McColm, G. L.; "*On the Power of Deterministic Closures"* Versión Final, Information and Computation, Vol. 119, 129-135, 1995"

[GMR99] Gagliardi, Edilma; Maldocena, Paulino y Reyes, Nora "*Utilización de Extensiones de Lógica de Primer Orden para la Computación de Queries en problemas de Redes"*; IV CACIC, Tandil, 1999.

[Imm87] Immerman, Neil; "*Languages that Capture Complexity Classes*"; SIAM Journal of Computing 16:4, 1987, 760-778.

[Pfa77] Pfaltz, John;*"Computer Data Structures*".McGraw-Hill-Kogakusha. 1977.

[Tur96] Turull Torres, José M.; "*Clases de Bases de Datos L-Rígidas y Expresividad de Lenguajes Relacionales Incompletos*". Tesis Doctoral, UNSL, 1996.

[Ull88] Ullman, Jeffrey D.; "*Principles of Database and Knowledge Base Systems*". Computers Science Press, 1988.