Computación de Queries a Bases de Datos Relacionales utilizando Circuitos Booleanos

Gagliardi, Edilma Olinda
Grosso, Alejandro Leonardo
Pereyra, Sonia Raquel
Piffaretti, Patricia
Turull Torres, José María *

{oli, agrosso, turull } @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

En este trabajo se muestra la utilización de Subfamilias Finitas de Circuitos Booleanos como un modelo teórico adecuado para la expresión de consultas a una Base de Datos Relacional, cuyo fundamento se basa en la equivalencia demostrada entre Lógica de Primer Orden y una clase restringida de familias de Circuitos Booleanos.

Se destaca un aspecto relevante que surge de utilizar los Circuitos Booleanos para la expresión de consultas a Bases de Datos Relacionales, puesto que constituyen un formalismo apropiado para apreciar la paralelizabilidad de las mismas. También se observa que al considerar aquellas aplicaciones en donde el dominio de la base de datos es fijo, y que se conoce que no se producirán alteraciones sobre la misma que le modifiquen, conforman casos apropiados para definir consultas a priori expresadas utilizando los Circuitos Booleanos, con el fin de implementarlas a nivel de hardware.

Se describe la implementación de un *traductor* de consultas expresadas en Lógica de Primer Orden a una Subfamilia Finita de Circuitos Booleanos para una Base de Datos Relacional dada; y la implementación de un *evaluador* de la consulta expresada como una Subfamilia Finita de Circuitos Booleanos.

PALABRAS CLAVES

Bases de Datos Relacionales, Circuitos Booleanos, Queries, Queries Computables, Lógica de Primer Orden, Cálculo Relacional, Cálculo Proposicional.

* Universidad Nacional de San Luis; Universidad Tecnológica Nacional (FRBA); Subsidio de Universidad CAECE

INTRODUCCIÓN

A partir de la traducción demostrada en [DENENBERG 86] entre Lógica de Primer Orden (FO) y una clase restringida de familias de Circuitos Booleanos de tamaño polinomial y profundidad constante (AC^0), es posible concebir la utilización de *Subfamilias Finitas de Circuitos Booleanos* [BALCAZAR 88] como un formalismo adecuado para la expresión de consultas a una Base de Datos Relacional. Allí se muestra cómo realizar la traducción de sentencias expresados en FO a Subfamilias Finitas de Circuitos Booleanos, cuando se considera la axiomatización de clases de estructuras finitas en FO. En el presente trabajo se extiende la traducción a la clase de todas las fórmulas bien formadas en FO, incluyendo no sólo sentencias sino también fórmulas con variables libres.

Un aspecto relevante que resulta al considerar los Circuitos Booleanos como un formalismo para la expresión de consultas a Bases de Datos Relacionales, es que constituyen un modelo teórico adecuado para apreciar la paralelizabilidad de la misma. Esto surge cuando se analiza la relación entre tamaño y profundidad de los circuitos que conforman la familia de circuitos que expresa la consulta y que puede influir en la utilización de computación paralela.

El desarrollo e implementación del presente trabajo consta de dos partes. La primera consiste de un *traductor* de consultas expresadas en FO (equivalente al Cálculo Relacional orientado a Dominios), a una Subfamilia Finita de Circuitos Booleanos representada en la codificación denominada Standard [BALCAZAR 88] para cada base de datos relacional dada. La segunda componente es un *evaluador*, en el cual dada una instancia de base de datos y la consulta expresada como una Subfamilia Finita de Circuitos Booleanos, computa la consulta correspondiente.

Los Circuitos Booleanos considerados aquí, son con grado de entrada limitado a una o dos entradas (*bounded* fan-in); y con grado de salida limitado a uno *(bounded* fan-out), obteniéndose Subfamilias de Circuitos Booleanos de *tamaño* polinomial y *profundidad* logarítmica. En el presente trabajo se muestra de qué manera los circuitos son construibles a partir de un algoritmo que muestra por si mismo la *uniformidad* [BALCAZAR 90].

En la primera sección se presentan los conceptos teóricos de interés; en la segunda sección se describe el desarrollo e implementación del presente trabajo; y en la última sección se presenta una aplicación que se ajusta correctamente a la utilización de los Circuitos Booleanos para la expresión de consultas a Bases de Datos Relacionales. Finalmente, se presentan conclusiones y proyecciones del presente trabajo.

SECCIÓN 1: ASPECTOS TEÓRICOS

1.1 INTRODUCCIÓN

Considerando una Base de Datos como una representación simbólica de una realidad acotada, de modo tal que el modelo sea finito, es deseable realizar consultas a la misma con el fin de obtener información, y para ello será suficiente con consultar el modelo. Por ello, es necesario contar con un lenguaje que permita formular las consultas (query/queries) a la Base de Datos. Para el caso del *Modelo Relacional*, modelo de interés para el presente trabajo, Codd [COD 70] desarrolló el *Álgebra Relacional (AR)* y el *Cálculo Relacional (CR)*, demostrando que ambos lenguajes son equivalentes en su poder expresivo, y más aún, equivalentes a la *Lógica de Primer Orden (FO)*[ULLMAN 88]. Observando el *Modelo Relacional* en el marco de la *Teoría de Modelos Finitos* [EBBINGHAUS 95], el esquema de la Base de Datos Relacional está dado por un vocabulario relacional finito σ, donde cada símbolo de relación en σ es de una cierta aridad; 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. De esta manera, se consideran a las Bases de Datos Relacionales como *Clases de Estructuras Relacionales Finitas*, y a las instancias de Bases de Datos Relacionales como *Estructuras Relacionales Finitas*.

Desde este marco, un query puede verse 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,

$f: \mathcal{E}_{\sigma, fin} \rightarrow \mathcal{E}_{<R>}$. En [CHANDRA 80] se propone como definición de la clase de queries computables 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. A dicha clase se la llamó CQ (Computable Queries).

Regresando al punto en donde se exponen los formalismos para expresar queries, se pueden considerar varios tipos de formalismos de diferentes poder expresivo, aquellos basados en máquinas abstractas(Máquina Genérica GM y GM loose, definidas por [ABITEBOUL 91]), otros en lógicas de diversos poderes expresivos(First Order, Extensiones de FO, Second Order, Lógicas Infinitarias), en lenguajes de programación formales(Query Language (QL)), o las familias de Circuitos Booleanos. Para el presente trabajo, el modelo de interés como formalismo apropiado para expresar y evaluar queries en Bases de Datos Relacionales son los Circuitos Booleanos.

Como se muestra en [DENENBERG 86] al considerar la axiomatización de clases de estructuras finitas en FO, la traducción es para sentencias expresadas en FO a Subfamilias Finitas de Circuitos Booleanos; en el presente trabajo se extiende tal traducción a la clase de todas las fórmulas bien formadas en FO, incluyendo no sólo sentencias sino también fórmulas con variables libres. Para ello, las mismas se traducen al Cálculo Proposicional (CP) [HAMILTON 81], y finalmente se genera la Subfamilia Finita de Circuitos Booleanos correspondiente. La traducción intermedia de FO a CP se puede realizar porque el dominio de la estructura es finito. Además, si se considera el árbol de expresión para una fórmula del CP, la misma es similar a la del grafo que representa un Circuito Booleano, cambiando conectivos lógicos por compuertas, y resultando, de esta forma, más natural la construcción del circuito.

Intuitivamente, la idea es construir, a partir de un query de aridad r y una Base de Datos cuyo dominio es de cardinalidad n, una Subfamilia Finita de Circuitos Booleanos $C=\{ C_0, C_1,\ldots ,C_q \}$ con $q= n^r -1$, donde cada C_i decide si la i-esima r-tupla pertenece o no al resultado del query considerándose un orden lexicográfico en las r-tuplas.

De esta forma, para cada n (cardinalidad de dominio), y fijando la aridad del query, se construye una Subfamilia Finita de Circuitos Booleanos. En consecuencia, la unión de todas las Subfamilias (es decir para cada n natural), constituye la Familia Infinita de Circuitos que computa un query (función) dado.

Los Circuitos Booleanos son un formalismo adecuado para apreciar la paralelizabilidad de la consulta. No siempre es cierto que el tiempo necesario para la evaluación de la consulta disminuya sustancialmente por el solo hecho de disponer de más de un procesador y utilizar procesamiento paralelo. En este sentido, es intuitiva la idea de que si se evalúa un circuito con una cantidad de procesadores en paralelo lo suficientemente grande como para cubrir el ancho máximo del mismo, el tiempo total para la computación del query estará determinado por la profundidad del circuito. La clase de las funciones "bien paralelizables" se formaliza con la clase $NC = \cup NC^i$, $i \geq 1$, donde NC^i = Size-Depth(n^c , $\log^i(n)$), es decir aquellas funciones que son computadas por una familia infinita de circuitos $C=\{ C_1, C_2,\ldots ,C_n, \ldots\}$, de tamaño polinomial, $O(n^c)$, y profundidad polilogarítmica, $O(\log^i(n))$. De modo que las clases NC^i representan lo que en general se consideran las clases de funciones para las que es conveniente contar con recursos de computación paralela.

Para la construcción de la familia de circuitos hay que dar un criterio de computabilidad, que se denomina *Uniformidad* [BALCAZAR 90] y que además provea un criterio para acotar los recursos necesarios para la construcción del circuito. En el presente trabajo se muestra de qué manera los circuitos son construibles a partir de un algoritmo que muestra por si mismo la Uniformidad.

1.2 BASES DE DATOS Y QUERIES

Sea $A = \{ (,),,, \wedge, \vee, \neg, \forall, \exists, x_1, x_2, \ldots x_n, \ldots\}$ un alfabeto infinito numerable. Sea σ un vocabulario relacional finito, $\sigma = < R_1, R_2, \ldots, R_k >$, el cual tiene definido los símbolos de relación juntamente con las aridades. 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 [EBBINGHAUS 84] .

Se dice que B es una σ-estructura finita definida como una k+1-tupla, denotada $B=\langle D^B, R_1^B, \ldots, R_k^B \rangle$, donde D^B es un conjunto finito, $D^B=\{d_1,...d_n\}$, denominado *Dominio* de B; y las relaciones $R_1^B, R_2^B, \ldots, R_k^B$ definidas en D^B, de la aridad adecuada y únicas para cada símbolo de relación del vocabulario σ, queriendo significar con ello que si B es una interpretación para el lenguaje L_σ de dominio D^B, la interpretación en B de R_i se denotará por R_i^B. Se asume la relación de Igualdad como parte del vocabulario.

Así, el esquema de la Base de Datos Relacional [TURULL 96] [EBBINGHAUS 95] 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.

Si σ es un vocabulario relacional finito, $\varphi \in L_\sigma$ con variables libres $\{x_1,x_2,...,x_r\}$, y B es una σ-estructura, con $d_1,\ldots,d_n \in D^B$, se denotará con la siguiente expresión ($\models$) al valor de verdad Verdadero de la fórmula φ interpretada en la estructura B, con el elemento d_{s_j} asignado a la variable libre x_j, para $1 \leq s_j \leq n$, $1 \leq j \leq r$, entonces $B \models \varphi(x_1, \ldots, x_r)[d_{s_1}, \ldots, d_{s_r}]$.

La acción de asignar el elemento d_{s_j} a la variable libre x_j, siendo x_j variable libre, es denominda *valoración.*

Sea $\varepsilon_{\sigma,fin}$ el conjunto de las estructuras relacionales finitas de un cierto vocabulario σ y sea $\varepsilon_{<R>}$ el conjunto de las relaciones definidas en el dominio de la estructura correspondiente, para alguna aridad r. Entonces, un query r-ario, para un entero r>0 , es una función de la forma $f: \varepsilon_{\sigma,fin} \rightarrow \varepsilon_{<R>}$, para algún vocabulario relacional σ, donde la aridad de *R* es r, y donde para toda estructura $B \in \varepsilon_{\sigma,fin}$, *f*(B) está definida en D^B. De modo análogo se denomina query 0-ario, o query booleano, a toda función de la forma $f: \varepsilon_{\sigma,fin} \rightarrow \{0,1\}$.

Se dice que *f* 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, es decir, para todo par de σ-estructuras I_1 e I_2, y para todo isomorfismo $h: I_1 \rightarrow I_2$ debe ser $f(I_2) = h(f(I_1))$[CHANDRA 80].

Nótese que de acuerdo a la semántica del lenguaje considerado, si φ es tal como se definió arriba, define o expresa un query r-ario f_φ sobre la estructura B. Es decir, f_φ (B), denotado por φ^B, es una relación finita definida en la estructura B por la fórmula φ, cuya aridad estará determinada por la cantidad de variables libres de φ. En símbolos:

$$\varphi^B = \{(d_{s_1}, \ldots, d_{s_r}) : d_{s_1}, \ldots, d_{s_r} \in D^B \wedge B \models \varphi(x_1, \ldots, x_r)[d_{s_1}, \ldots, d_{s_r}], 1 \leq s_j \leq n, 1 \leq j \leq r\}$$

1.3 CIRCUITOS BOOLEANOS

Un circuito booleano [BALCAZAR 88], [BALCAZAR 90] puede verse como un grafo dirigido acíclico (DAG), el cual se construye asociando a cada nodo rotulado una variable, una constante o una compuerta booleana ($\wedge,\vee,\neg$), y uniendo el nodo g_i al nodo g_j con un arco, si la salida de la compuerta g_i es una entrada a g_j .

Dado un circuito C se define como tamaño de C, Size(C), a la cantidad de compuertas de C y como profundidad de C, Depth(C), a la longitud del camino más largo en C, desde las compuertas de entrada a la compuerta de salida final.

Los circuitos poseen una cantidad finita de entradas, tal que cuando se utilizan para computar una función $f:\{0,1\}^* \rightarrow \{0,1\}$ de dominio infinito, se debe construir un circuito diferente para cada longitud de los elementos del dominio de f. Así se da origen a una familia infinita de circuitos $C=\{C_0, C_1, \ldots, C_n, \ldots\}$, donde cada C_n computa la función restringida al dominio $\{0,1\}^n$.

Los Circuitos Booleanos aquí utilizados tienen grado de entrada limitado a una o dos entradas, y grado de salida limitado a una única salida, obteniéndose Subfamilias de Circuitos Booleanos de tamaño polinomial y profundidad logarítmica. Esto es O(n) y O(log(n)) respectivamente. En consecuencia, las funciones computadas por estos Circuitos Booleanos pertenecen a las clases de funciones bien paralelizables. Obsérvese que, con estas restricciones, no hay reuso de subcircuitos.

Una codificación standard, $\overline{C}$, de un circuito *C*, es una secuencia de cuádruplas $q_1,..., q_k$ de la forma $q_h = (g, b, g_i, g_d)$, $1 \leq h \leq k$; donde g representa el número de compuerta; b es la operación booleana de la compuerta; g_i es el número de la compuerta, variable o constante, que provee la entrada izquierda a g , y g_d es el número de compuerta, variable o constante que provee la entrada derecha a g. No se codifican cuádruplas para los nodos rotulados con variables de entrada. Se toma la convención de que la compuerta de salida de un circuito C es la compuerta definida en la última cuádrupla de la secuencia de cuádruplas que define al circuito.

Intuitivamente, la idea es construir a partir de un query de aridad r, la Subfamilia Finita de Circuitos Booleanos $C=\{C_0,C_1,...,C_q\}$, con $q = n^r-1$, siendo n la cardinalidad del dominio de la base de datos, donde cada C_i con r entradas y una salida decide si la i-ésima r-tupla, considerando el orden lexicográfico de las r-tuplas, pertenece o no al resultado del query. Nótese que cada circuito C_i evalúa la i-ésima r-tupla y decide si la misma pertenece o no al resultado.

Como se mencionó anteriormente, los Circuitos representan un modelo apropiado para el análisis de paralelismo. En una Subfamilia Finita de Circuitos Booleanos $C=\{C_0,C_1,...,C_q\}$, se puede considerar explotar el paralelismo en dos sentidos. Uno, a nivel del circuito propiamente dicho, manteniendo tantos procesadores como fuera necesario para cubrir el ancho máximo del mismo (wide). Y otro, sentido sería a nivel de la familia, manteniendo procesadores que trabajen en paralelo, cada uno de ellos evaluando un circuito C_i de la familia C.

Se puede apreciar que si la familia de circuitos que computa el query pertenece a NC^i, el tiempo necesario para computar el mismo puede disminuir de tiempo polinomial a polilogarítmico cuando se cambia de computación secuencial a paralela, manteniendo una cantidad polinomial de procesadores, lo cual es considerado como "factible" en Teoría de Complejidad Paralela.

1.4 TRADUCCIÓN DE UNA FÓRMULA DE FO A CIRCUITOS BOOLEANOS

Al interpretar una fórmula atómica se puede decir si ésta se satisface o no en la interpretación para una valoración dada, lo que equivale a decir si es verdadera o falsa. Considerando la semántica de la fórmula atómica en una valoración, se la puede observar como una proposición, y asociarle por lo tanto una variable proposicional que la represente. Se dice entonces, que la variable proposicional es valorada con el valor de verdad Verdadero si la fórmula en la correspondiente valoración se satisface, y con el valor de verdad Falso en otro caso. De esta manera, las fórmulas atómicas expresadas en FO y valoradas, pueden ser traducidas al Cálculo Proposicional (CP).

Considerése el vocabulario $\sigma =< R_1,R_2,\ldots,R_k >$, tal que a_i es la aridad para cada R_i respectivamente; sea la σ-estructura $B=\langle D^B, R_1^B,\ldots,R_k^B\rangle$, con $D^B=\{d_1,d_2,...,d_n\}$ y $|D^B|=n$; y sea $R_h(x_1,\ldots,x_{a_h})$ una fórmula atómica cualquiera de L_σ, con $1 \leq h \leq k$. Se observará entonces que al interpretar $R_h(x_1,\ldots,x_{a_h})$ con una valoración dada en la σ-estructura B, quedará una a_h-tupla que puede ser considerada como una variable proposicional p. Si la a_h-tupla resultante en esa valoración pertenece a R_h^B, la proposición p tendrá el valor de verdad V, sino tendrá el valor de verdad F. Sea *v* una valoración cualquiera, entonces $B \models R_h(x_1,\ldots,x_r)$ si y sólo si $R_h^B(v(x_1),\ldots, v(x_r))$ se cumple, es decir $(d_{s_1},\ldots,d_{s_r}) \in R_h^B$, con $1 \leq s_j \leq n$, $1 \leq j \leq r$. Luego, la cantidad de valoraciones obtenidas en una relación de aridad r con un dominio de cardinalidad n, es $|D^B|^r=n^r$. De modo que asociando a cada valoración una variable proposicional, se obtienen n^r variables proposicionales, y cada una de ellas puede tomar un valor de verdad Verdadero o Falso, dependiendo de la satisfacción o no de la fórmula atómica.

Por lo analizado anteriormente, se ve claramente que a cada fórmula atómica en una valoración particular se le puede asociar una variable proposicional y que la cantidad de variables proposicionales dependen de la aridad de la relación y la cardinalidad del dominio.

Entonces si la cardinalidad del dominio es n, la numeración de todas las variables proposicionales se hace siguiendo el orden consecutivo de las relaciones del vocabulario, considerando que para cada relación de aridad r, le corresponden n^r variables proposicionales. Por lo tanto, dada la relación $R_i^{(ai)}$ le corresponden n^{ai} variables proposicionales, numeradas desde el número $h= \Sigma_{j<i}\, n^{aj}$ hasta el número $(h+n^{(ai)}-1)$.

Siguiendo a [DENENBERG 86], dada una sentencia φ en FO se puede construir el circuito C que representa a la fórmula de manera inductiva, construyendo un único nodo con un arco rotulado φ. Si a su vez $\varphi \equiv \varphi_1 \wedge \varphi_2$, éste será un DAG con un nodo rotulado $\wedge$, y como entradas los arcos rotulados φ_1 y φ_2 respectivamente; para los casos $\varphi \equiv \varphi_1 \vee \varphi_2$ y $\varphi \equiv \neg\varphi_1$, la representación en DAG es similar. Si $\varphi \equiv \forall x\ \varphi_1(x)$, el DAG correspondiente es un nodo rotulado $\wedge$, con n arcos de entrada rotulados $\varphi_1(x)[d_1],\ldots, \varphi_1(x)[d_n]$ respectivamente, con $d_1,...,d_n \in D^B$, siendo D^B el dominio de la interpretación. Si $\varphi \equiv \exists x\ \varphi_1(x)$ la construcción es similar, sólo que tendrá el nodo rotulado con el operador booleano $\vee$. Continuando con la descomposición de φ en subfórmulas, se llega al punto en que φ es una composición de fórmulas atómicas, las cuales valoradas en la interpretación, serán las variables proposicionales.

Considerando el DAG resultante con la codificación standard $\overline{C}$ se obtiene la secuencia de cuádruplas que lo representan.

Basándose en [HAMILTON 81], [EBBINGHAUS 84], [DENENBERG 86], [BALCAZAR 88], se dan a continuación las reglas de traducción.

Dados σ un vocabulario relacional finito, B una σ-estructura con $d_1,\ldots,d_n \in D^B$, y v una valoración, se muestra cómo $\varphi \in L_\sigma$ es traducida a CP y expresada finalmente en $\overline{C}$:

Primer Orden	***Cálculo Proposicional***	***Circuito Booleano***
Átomo φ_1		
(1) Si φ_1 es un átomo y subfórmula propia de φ		
$R(x_1,..,x_r)\ [(v(x_1),..,.v(x_r))]$	p variable proposicional	Entrada al circuito (g_i o g_d)

Obsérvese: Para las traducciones dadas a continuación se hace uso de las siguientes notaciones:

- g_{φ_i} denota la salida de la compuerta que representa a la fórmula φ_i
- $g_{\varphi_j(d_i)}$ denota la salida de la compuerta que representa a la fórmula $\varphi_j(x)[d_i]$

Primer Orden	***Cálculo Proposicional***	***Circuito Booleano***
Fórmulas Bien Formadas: Sean φ_1 y φ_2 fórmulas bien formadas y x una variable de individuo.		
(2) φ es un átomo $\varphi \equiv R(x_1,..,x_r)[(v(x_1),..,.v(x_r))]$	p es variable proposicional	$(g, \wedge, p, p)$
(3) $\varphi \equiv \neg\varphi_1$	$\neg\varphi_1$	$(g, \neg, nil, g_{\varphi_1})$
(4) $\varphi \equiv (\varphi_1\ \theta\ \varphi_2)\ \theta \in \{\wedge, \vee\}$	$(\varphi_1\ \theta\ \varphi_2)$	$(g, \theta, g_{\varphi_1}, g_{\varphi_2})$
(5) $\varphi \equiv \forall(x)\varphi_1$	$\varphi_1(x)[d_1] \wedge \ldots \wedge \varphi_1(x)[d_n]$	$(g_1, \wedge, g_{\varphi_1(d_1)}, g_{\varphi_1(d_2)})$

$(g_2, \wedge, g_1, g_{\varphi_1(d_3)})$

...

$(g_{n-1}, \wedge, g_{n-2}, g_{\varphi_1(d_n)})$

(6) $\varphi \equiv \exists (x) \varphi_1$ ídem paso (5) sólo que con el operador booleano $\vee$ como tipo de compuerta.

Obsérvese que en los puntos (5) y (6) se muestra una cadena de cuádruplas que representan a un circuito cuya profundidad es lineal, O(n), sólo a los efectos de clarificar la expansión que implican los cuantificadores.

1.5 CONSTRUCCIÓN DE LA SUBFAMILIA FINITA DE CIRCUITOS BOOLEANOS

En base a lo presentado, se verá a continuación cómo generar la subfamilia correspondiente al query [PEREYRA 98].

Dada la Base de Datos $B=\langle D^B, R_1^B, \ldots, R_k^B \rangle$, con $D^B=\{d_1,d_2,...,d_n\}$, $|D^B|$ =n cabe preguntar ¿cuántos circuitos son necesarios para poder expresar un query? o lo que es equivalente, ¿cuántas valoraciones posibles hay para $\varphi(x_1, \ldots, x_r)$ si el dominio de la interpretación es de cardinalidad n? La respuesta a esto es todas las posibles combinaciones de las r variables libres en D^B, que está definido por n^r.

De modo que si:

a) r>0 (query r-ario), expresado en FO como $\varphi(x_1, \ldots, x_r)$, con r variables libres, cada valoración v_i para las variables libres $\{ x_1, \ldots, x_r. \}$ de φ, en el dominio D^B, genera un circuito C_i . Con d_{kp} se denota colocar en la posición p de la tupla el elemento k; es decir:

$C_0 \equiv \varphi(x_1, \ldots, x_r)[d_{11}, \ldots, d_{r1}] \equiv \varphi_0$	valoración de las r variables libres en el primer elemento del dominio.
$C_1 \equiv \varphi(x_1, \ldots, x_r)[d_{11}, \ldots, d_{r2}] \equiv \varphi_1$	Valoración de las r-1 primeras variables libres en el primer elemento del dominio y la r-ésima variable libre en el segundo elemento.
.....	.....
$C_q \equiv \varphi(x_1, \ldots, x_r)[d_{1n}, \ldots, d_{rn}] \equiv \varphi_q$	valoración de las r variables libres en el enésimo elemento del dominio

Para un query r-ario se tiene que $q= n^r -1$, entonces se obtiene una Subfamilia Finita de Circuitos Booleanos $C=\{C_0,C_1,...C_q\}$ donde cada C_i , es representado por una secuencia finita de cuádruplas $q_1,q_2,\ldots,q_k$. Entonces, un query r-ario se convierte en n^r queries booleanos del siguiente modo:

Para cada valoración $v_i : \{ x_1, \ldots, x_r. \} \rightarrow D^B$, el query booleano φ_i , es $\varphi(x_1, \ldots, x_r)[v_i(x_1),..., v_i(x_r)]$, $0 \leq i \leq q$.

b) r=0, (Query 0-ario), expresado como una sentencia φ de FO, genera un único circuito C, dado que $n^0=1$.

De esta manera se construye la Subfamilia Finita de Circuitos Booleanos que representan a $\varphi(x_1, \ldots, x_r)$. El siguiente paso consiste en evaluar dicha subfamilia.

1.6 EVALUADOR

Si σ es un vocabulario relacional finito, B es una σ-estructura, con $d_1,\ldots,d_n \in D^B$, y $\varphi \in L_\sigma$ con { $x_1, \ldots,x_r$ } variables libres, se denotará con la siguiente expresión al valor de verdad Verdadero de la fórmula φ interpretada en la estructura B, con el elemento d_{s_j} asignado a la variable libre x_j, para $1 \leq s_j \leq n$, $1 \leq j \leq r$, $B \models \varphi(x_1, \ldots,x_r)[d_{s_1},\ldots,d_{s_r}]$.

Denotamos con φ^B, indistintamente, a la fórmula y a la relación que ella expresa:

$\varphi^B = \{(d_{s_1},\ldots,d_{s_r}) : d_{s_1},\ldots,d_{s_r} \in D^B \wedge B \models \varphi(x_1, \ldots,x_r)[d_{s_1},\ldots,d_{s_r}], 1 \leq s_j \leq n, 1 \leq j \leq r\}$

A través de la equivalencia de representación, una fórmula $\varphi \in L_\sigma$ con { $x_1, \ldots,x_r$ } variables libres, interpretada en B para alguna valoración dada, es ahora representada en el formalismo de Circuitos Booleanos por un circuito C_h, para algún h dado ($\equiv \varphi_h$), y se denotará con $C_h(B)$ al valor de verdad de C_h interpretado en la estructura B, con el elemento d_{s_j} asignado a la variable libre x_j con $1 \leq s_j \leq n$, $1 \leq j \leq r$. Si ocurre que $B \models \varphi(x_1, \ldots,x_r)[d_{s_1},\ldots,d_{s_r}]$ y como $C_h \equiv \varphi(x_1, \ldots,x_r)[d_{s_1},\ldots,d_{s_r}] \equiv \varphi_q$, entonces φ^B podría verse aquí como:

$\varphi^B = \{(d_{s_1},\ldots,d_{s_r}) : d_{s_1},\ldots,d_{s_r} \in D^B \wedge C_h(B) \equiv$ Verdadero, $q = n^r - 1$, $0 \leq h \leq q$, $1 \leq s_j \leq n$, $1 \leq j \leq r\}$

Obsérvese que h está ligado a la r-tupla particular $(d_{s_1},\ldots,d_{s_r})$

El evaluador, entonces da como resultado el conjunto de r-tuplas para las cuales cada $C_h(B) \equiv$ Verdadero; en el caso en que r=0, φ una sentencia, la respuesta es Sentencia Verdadera o Sentencia Falsa.

SECCIÓN 2: TRADUCTOR Y EVALUADOR

2.1 INTRODUCCIÓN

El *traductor* tiene como entrada un query expresado en lenguaje de primer orden puramente relacional (FO) para una base de datos dada, y como salida la Subfamilia Finita de Circuitos Booleanos. El *evaluador* toma como entrada dicha Subfamilia Finita de Circuitos Booleanos y devuelve como resultado la relación cuyas nuplas satisfacen el query en tal base de datos [PEREYRA 98].

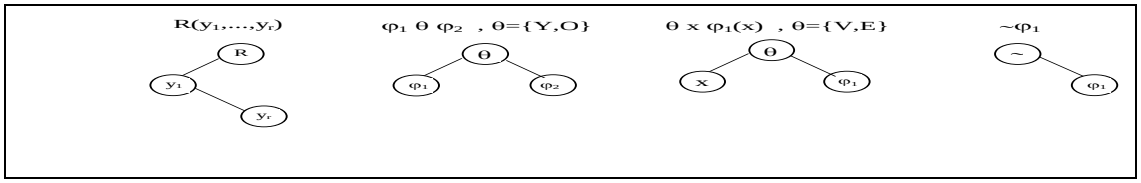
2.2 IMPLEMENTACIÓN

La instancia de la Base de Datos se ingresa por extensión y a partir de la misma se contruye el dominio, el esquema, la relación de igualdad, y la lista de variables proposicionales. Así dada la σ-estructura $B = < R_1, R_2, \ldots, R_K >$ juntamente con sus aridades $a = \{ a_1, a_2, \ldots, a_k \}$, donde a_i es la aridad de R_i respectivamente, $1 \leq i \leq k$. La numeración de las Variables Proposicionales está dada en el rango $[0\ldots (\sum_{i=1}^{k} |D^{a_i}|)-1]$.

Se define el query siguiendo las reglas de construcción sintácticas de FO, indicando su nombre, el conjunto de las *variables libres* y la *fórmula bien formada* que lo expresa. El alfabeto consta de los símbolos ∧, ∨ y ~ que representan las *conectivas de conjunción, disyunción y negación* respectivamente*;* los símbolos {(,), }, como *símbolos de puntuación;* los símbolos ∀, ∃ para los *cuantificadores universal y existencial* respectivamente.

2.2.1 Construcción del árbol de expresión para φ

Para la generación de la Subfamilia Finita de Circuitos Booleanos, en principio se toma la fórmula φ, se realiza el análisis sintáctico y se construye el árbol de expresión correspondiente de la siguiente forma: Si R (y_1, …,y_r) es una fórmula atómica, φ_1 y φ_2 son fórmulas bien formadas, e y_1, …,y_r , x son variables de individuo, se muestran los subárboles correspondientes para cada caso:



Se definen en este proceso nodos de distinto tipo que determinarán la acción semántica a seguir en la construcción del circuito.

2.2.2 Construcción de cada C_i

Una vez generado el árbol de expresión para φ , entonces para cada valoración v_i de φ en D^B, con n = | D^B |, se recorre el árbol de expresión de acuerdo a un recorrido postorder recursivo y se genera cada C_i . Cada C_i es una secuencia finita de cuádruplas q_1,..,q_p de la forma q_h= (g, b, g_i, g_d), $1 \leq h \leq p$, donde g representa el número de compuerta; b es la operación booleana de la compuerta; g_i es el número de la compuerta, variable, que provee la entrada izquierda a g; y g_d es el número de compuerta, variable que provee la entrada derecha a g. La numeración de las Variables Proposicionales está dada en el rango [0… ($\sum_{i=1}^{k} |D^{a_i}|$)-1], y se utiliza como primer número de compuerta a ($\sum_{i=1}^{k} |D^{a_i}|$), de manera que el sistema pueda diferenciar entre una variable proposicional y una compuerta.

Para las valoraciones se numeran los elementos del dominio en el rango [0…(| D^B |-1)], y de sta manera se referencian los mismos. Luego las valoraciones se realizan del siguiente modo: inicialmente todas las variables libres están en cero. Si x_1, x_2...,x_r son las variables libres, la cantidad de valoraciones que existen en D^B son $|D^B|^r$. Cada vez que una variable libre llega al último elemento del dominio, ésta se vuelve a instanciar en cero y la que le precede se incrementa en uno, procediendo así hasta completar todas las valoraciones. Por cada valoración se recorre el árbol recursivamente en postorder, dado que con este recorrido cada vez que visita un nodo es porque ya recorrió sus subárboles izquierdo y derecho, y por lo tanto tiene los resultados de su evaluación disponible para realizar la acción semántica que le requiere el tipo de nodo que se está tratando.

Acciones Semánticas:

a. Tipo de nodo Variable Libre: Sea x la variable libre. Se le asigna el elemento del dominio que le corresponde según la valoración v_i .

b. Tipo de nodo Relación (R): En este punto del recorrido, se tiene la fórmula atómica R con todos sus términos valorados, generando la variable proposicional p correspondiente, de acuerdo al orden establecido para las fórmulas atómicas sin variables, la cual será la entrada del circuito C_i.

c. Tipo de nodo Conectivas {∧,∨ }: En el análisis de una conectiva binaria, está resuelta su entrada izquierda g_i y su entrada derecha g_d , generando una de las siguientes cuádruplas: (g,∧, g_i, g_d) o (g, ∨, g_i, g_d) según sea el caso, donde *g* es el número que se le asigna a la compuerta ∧ o ∨.

*d. Tipo de nodo Conectiva ~ :*La cuádrupla generada es (g, ~, nil, g_d), donde nil representa que la compuerta g no tiene entrada izquierda.

e. Tipo de nodo Definición de variable ligada: Cuando se encuentra una variable ligada, significa que se comienza a analizar el *radio de acción* de un *cuantificador* , $\forall$ *o* $\exists$, en un nuevo elemento del dominio, por ejemplo d_i ($1 \leq i \leq n$), significando que toda aparición de la variable ligada en el radio de acción del cuantificador será valorada en ese elemento.

f. Tipo de nodo Asignación de variable ligada: Este nodo corresponde a un nodo del subárbol derecho del cuantificador y representa a la variable ligada que se encuentra durante el análisis de la subexpresión φ_1. La acción semántica que se implementa en este punto recupera el valor de la variable ligada.

g. Tipo de nodo Cuantificador { $\forall$,$\exists$}: Cada cuantificador expande n ($n=|D^B|$) veces la subfórmula, uniendo las sub-subfórmulas respectivas por la operación booleana correspondiente (conjunción o disyunción). El tamaño del circuito resultante dependerá de n y de la complejidad de la subfórmula.

Sea $\forall z\, \psi(z)$ la subfórmula que se analiza. Si la variable ligada se valoró en todos los elementos del dominio, significa que se han generado los n subcircuitos de igual tamaño y profundidad, correspondiente a cada valoración de la variable ligada en los elementos del dominio; es decir, las subfórmulas $\psi(x)[d_i]$ ($1 \leq i \leq n$). Se genera, entonces, el subcircuito que representa al cuantificador, que en el caso del cuantificador universal $\forall$ será la conjunción de los $\psi(x)[d_i]$, y para el caso del cuantificador existencial $\exists$ será la disyunción de los $\psi(x)[d_i]$ ($1 \leq i \leq n$). Para esta generación se utiliza el algoritmo AND-TREE [BALCAZAR 90].

2.2.3 Evaluador

El evaluador toma como entradas la lista de variables proposicionales valoradas en Verdadero o Falso, y la Subfamilia Finita de Circuitos Booleanos $C=\{C_0, C_1, ..., C_q\}$, con $q = |D^B|^r - 1$, que representa el query $\varphi(x_1, \ldots, x_r)$, y devuelve como resultado el conjunto de tuplas $(v(x_1), \ldots, v(x_r))$, tal que $B \models \varphi(x_1, \ldots, x_r)[v(x_1), \ldots, v(x_r)]$ para alguna valoración *v*; es decir que la evaluación del circuito booleano correspondiente a tal valoración *v*, toma el valor de verdad Verdadero.

Dada una cuádrupla (g, op, g_i, g_d), sea $vv(g_i)$ el valor de verdad de la entrada g_i a g y $vv(g_d)$ el valor de verdad de la entrada g_d a g.

Para realizar la evaluación de *C* es necesario evaluar cada C_i con i= 0..q. Por cada C_i , se toma la cuádrupla de salida como nodo minimal del árbol de expresión que representa a C_i , y se realiza un recorrido postorder del circuito. Para cada cuádrupla perteneciente a C_i , el valor de verdad en la salida de la compuerta, se obtiene aplicando recursivamente las siguientes reglas de evaluación:

1º.- Entrada Izquierda a la compuerta g:
- g_i es nil, entonces evalúa la entrada derecha;
- g_i es un número de compuerta, entonces evalúa primero el subcircuito cuyo minimal es g_i.
- g_i es una variable proposicional (nodo maximal) entonces toma su valor de verdad de la lista de variables proposicionales y éste es $vv(g_i)$.

2º.- Entrada Derecha a la compuerta g:
- g_d es un número de compuerta, entonces evalúa primero el subcircuito cuyo minimal es g_d .
- g_d es una variable proposicional (nodo maximal) entonces toma su valor de verdad de la lista de variables proposicionales y éste es $vv(g_d)$.

3º.- Valor de salida de g:
- $vv(g_i)\ \theta\ vv(g_d)$ con $\theta \in \{\wedge, \vee\}$.
- $\sim vv(g_d)$

4º.- nada más

Sea C_i el circuito evaluado, entonces si el valor de verdad de la compuerta de salida es Verdadero, se debe dar como respuesta del query la tupla $(v(x_1), \ldots, v(x_r))$ correspondiente.

Para saber cuál es la valoración *v* que corresponde a C_i , se expresa el número i en base n, esto es i_n , el cual tiene r dígitos $dig_{s_1}, ..., dig_{s_r}$, $0 \leq s_j \leq (n-1)$, $1 \leq j \leq r$, tal que la variable libre x_j, $1 \leq j \leq r$, fue valorada con el elemento del dominio $d_{dig_{s_j}+1}$ perteneciente a D^B.

Ejemplo:

Sea $D^B = \{ d_1, d_2, d_3 \}$, $|D^B| = 3$, el query es $\varphi(x_1,x_2)$, cuya aridad es r = 2,y sea C_I con i= 7 el circuito cuyo resultado es verdadero. Entonces se quiere conocer cuál es la valoración *v* para x_1 y x_2 correspondiente al circuito C_7 que satisface al query. Entonces se expone i en base 10 y 3, esto es: $i_{10}=7_{10} = 21_3 = i_3$. De i expresado en base 3 ($|D^B|$), se toman los dígitos s_1 y s_2 que componen tal número, $dig_{s_1} = 2$, $dig_{s_2} = 1$. Se obtiene la valoración *v* correspondiente al circuito C_7, es decir $v(x_1)= d_{2+1} = d_3$ y $v(x_2) = d_{1+1} = d_2$. Entonces cuando x_1 vale d_3 y x_2 vale d_2 el query se satisface.

SECCIÓN 3: PRESENTACIÓN DE UNA APLICACIÓN

Considérese una base de datos que represente imágenes de igual tamaño en pixels, con la relación de vecindad adyacente entre pixels, y el color como atributo de pixel. Modelando la imagen como una matríz de n-filas por m-columnas, los pixels pueden nombrarse subindicándolos por número de fila y número de columna. Sin pérdida de generalidad, dado un pixel $Pix_{i\,j}$, su vecindad adyacente está formada por los pixels $Pix_{i-1\,j-1}$, $Pix_{i-1\,j}$, $Pix_{i-1\,j+1}$, $Pix_{i\,j-1}$, $Pix_{i\,j+1}$, $Pix_{i+1\,j-1}$, $Pix_{i+1\,j}$ y $Pix_{i+1\,j+1}$.

$Pix_{i-1\,j-1}$	$Pix_{i-1\,j}$	$Pix_{i-1\,j+1}$
$Pix_{i\,j-1}$	Pix_{ij}	$Pix_{i\,j+1}$
$Pix_{i+1\,j-1}$	$Pix_{i+1\,j}$	$Pix_{i+1\,j+1}$

Entonces se tiene:

- **Vocabulario** $\sigma = < Pixels^{(1)}, Color^{(1)}, Pixels\text{-}vecinos^{(2)}, Color\text{-}por\text{-}pix^{(2)}, Igual^{(2)} >$
- **Base de Datos o σ-estructura B** = $<D^B, Pixels^B, Colores^B, Pixels\text{-}vecinos^B, Color\text{-}por\text{-}pix^B, Igual^B >$
- **Dominio** de la estructura $\mathbf{D^B} = \{pix_{11}, pix_{12}, ..., pix_{nm}, col_1,..., col_k \}$, con $\mathbf{N= |D^B| = n*m + k}$, con k, n y m arbitrarios, y sea tal cardinalidad nombrada **N**
- **Instancias de las relaciones**
  - **$Pixels^B$ =** $\{pix_{11}, pix_{12}, ..., pix_{nm}\}$ donde $|\,Pixels^B\,| = n*m$
  - **$Colores^B$ =** $\{col_1, col_2, ..., col_k\}$ donde $|\,Colores^B\,| = k$
  - **$Pixels\text{-}vecinos^B$** = {(x,y) / x,y ∈ $Pixels^B$ *y* "el pixel x tiene de vecino adyacente al pixel y"}
    = $\{(pix_{11}, pix_{12}),(pix_{11},pix_{21}), (pix_{11}, pix_{22}), ..., (pix_{nm}, pix_{n\,m-1}),$
    $(pix_{nm}, pix_{n-1\,m-1}), (pix_{nm}, pix_{n-1\,m}) \}$

    Dado que es una relación simétrica se mantiene el generador minimal de la misma, el cual es una relación minimal en cuanto a la cantidad de tuplas, y que a partir de ella y de la propiedad simétrica se reconstruye la relación real.
  - **$Color\text{-}por\text{-}pix^B$** = {(x,y) / x ∈ $Colores^B$, y ∈ $Pixels^B$ *y* "y posee el color x" }
    = $\{(col_i , pix_{11}),(col_j ,pix_{12}), ..., (col_k ,pix_{nm}) \}$

    Esta relación es completa respecto de Pixels. Es decir, la proyección sobre la segunda componente se obtiene la relación $Pixels^B$.
  - **$Igual^B$** = {(x,x) / x ∈ D^B } = $\{(pix_{11}, pix_{11}) ,..., (col_k,col_k) \}$ donde $|\,Igual^B\,| = N$
    Esta relación es completa respecto de D^B .
- **Variables Proposicionales**

A partir del dominio D^B y de la σ-estructura B se obtienen las variables proposicionales. Sean $N= |D^B|$ y $L = N^2$.

- A la relación Pixels le corresponden las primeras N^1 variables proposicionales: (P_0... P_{N-1})
- A la relación Colores le corresponden las N^1 variables proposicionales siguientes: (P_N... P_{2N-1})

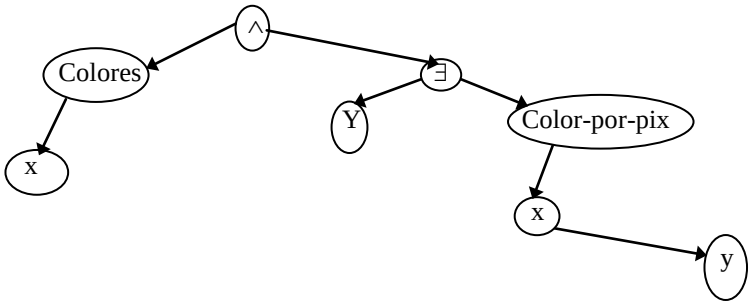
- A la relación Pixels-Vecinos, las $L = N^2$ variables proposicionales siguientes: (P_{2N}... $P_{2N+\ L-\ 1}$)
- A la relación Color-por-Pix , las $L = N^2$ variables proposicionales siguientes: (P_{2N+L} ...$P_{2N+\ 2L\ -1}$)
- A la relación Igual le corresponden las $L = N^2$ variables proposicionales siguientes: ($P_{2N+\ 2L}$ …$P_{2N\ +\ 3L-\ 1}$)

Obsérvense a continuación las variables proposicionales de los respectivos enunciados con la numeración acorde, y supóngase una asignación de valor de verdad para los mismos:

Nº de Variable Proposicional	Relación con términos instanciados en cada elemento del dominio D	Asignación de valor de verdad a cada proposición
P_0	Pixels(pix_{11})	
...	**…**	
P_{N-1}	Pixels(col_k)	
P_N	Colores(pix_{11})	Se asigna el valor de verdad *Verdadero* o *Falso* a cada variable de proposición de acuerdo a la σ-estructura B
...	**…**	
P_{2N-1}	Colores(col_k)	
P_{2N}	Pixels-vecinos(pix_{11},pix_{11})	
...	**…**	
$P_{2N+N-1} \equiv P_{3N-1}$	Pixels-vecinos(pix_{11},col_k)	
...	**…**	
$P_{N\ +\ L}$	Pixels-vecinos(col_k,pix_{11})	
...	**…**	
P_{2N+L-1}	Pixels-vecinos(col_k,col_k)	
P_{2N+L}	Color-por-pix(pix_{11},pix_{11})	
...	**…**	
P_{3N+L-1}	Color-por-pix(pix_{11},col_k)	
	…	
$P_{N+\ 2L}$	Color-por-pix(col_k,pix_{11})	
...	**…**	
$P_{2N+2L-1}$	Color-por-pix(col_k,col_k)	
P_{2N+2L}	Igual(pix_{11}, pix_{11})	
...	**…**	
$P_{2N+3L-1}$	Igual(col_k, col_k)	

- **Consulta a la base de datos**

A continuación sea la **Consulta** de interés**:** ***Colores que aparecen en una imagen.***
Para ello sea el **Query** expresado en Cálculo Relacional: $\varphi(x) \equiv Colores(x) \wedge \exists y\ Color\text{-}por\text{-}pix(x, y)$
Con el siguiente grafo obsérvese el **Árbol de expresión** para φ (x)



- **Traducción de φ al CP**

Se valora la variable x para todos los elementos del dominio, obteniendo φ_0, φ_1, … , φ_q respetivamente, con $\mathbf{q=N^r-1}$.

Valorando x en el elemento pix_{11} :

$\varphi(x)[pix_{11}] \equiv Colores(x)[pix_{11}] \wedge$
$(Color\text{-}por\text{-}pix\ (x,y)[pix_{11},pix_{11}]$
$\vee Color\text{-}por\text{-}pix(x,y)[pix11,pix_{12}]$
$\vee \ldots \vee Color\text{-}por\text{-}pix(x,y)[pix11,col_k])$

$\varphi(x)[pix_{11}] \equiv \boldsymbol{\varphi_0}$

Luego haciendo la correspondencia con las variables proposicionales se obtiene:

$\boldsymbol{\varphi(x)[pix11] \equiv P_N \wedge (P_{2N+L} \vee P_{2N+L+1} \vee \ldots \vee P_{3N+L-1}) \equiv \varphi_0}$

Así siguiendo para todos los elementos del dominio, hasta el último:
Valorando x en el elemento col_k :

$\varphi(x)[col_k] \equiv Colores(x)[col_k] \wedge$
$(Color\text{-}por\text{-}pix\ (x,y)[col_k,pix_{11}]$
$\vee Color\text{-}por\text{-}pix(x,y)[col_k,pix_{12}]$
$\vee \ldots \vee Color\text{-}por\text{-}pix(x,y)[col_k,col_k])$

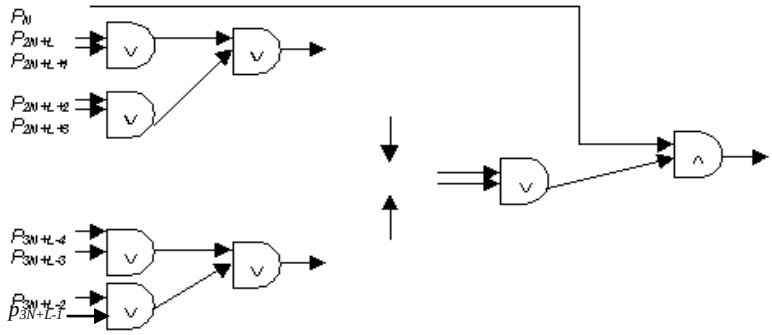
$\varphi(x)[col_k] \equiv \boldsymbol{\varphi_q}$

Luego haciendo la correspondencia con las variables proposicionales se obtiene:

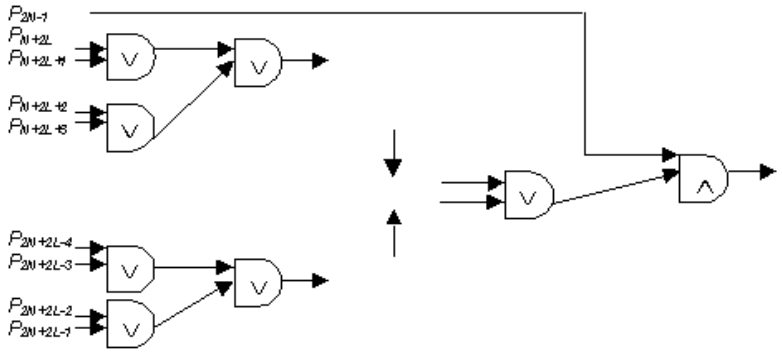
$\boldsymbol{\varphi(x)[col_k] \equiv P_{2N-1} \wedge (P_{N+2L} \vee P_{N+2L+1} \vee \ldots \vee P_{2N+2L-1}) \equiv \varphi_q}$

- **Traducción del CP a la Subfamilia de Circuitos Booleanos**

Obsérvese que el Circuito $\boldsymbol{C_0}$ corresponde a la $\boldsymbol{\varphi_0}$, tal que $\boldsymbol{C_0 \equiv \varphi_0}$. Cada circuito se construye siguiendo la metodología presentada en la sección anterior.



Obsérvese que el Circuito $\boldsymbol{C_q}$ corresponde a la $\boldsymbol{\varphi_q}$, tal que $\boldsymbol{C_q \equiv \varphi_q}$



- **Evaluación**

Tomando la lista de variables proposicionales valoradas en Verdadero o Falso, según la asignación que corresponda de acuerdo a la σ-estructura B, y la Subfamilia Finita de Circuitos Booleanos $C=\{C_0,C_1,...,C_q\}$, con $q = (|D^B|^r - 1)$, que representa al query $\varphi(x)$, el evaluador devuelve como resultado el conjunto de tuplas $(v(x))$ para alguna valoración v, tal que $B \models \varphi(x)[v(x)]$ se satisface; es decir que toma el valor de verdad Verdadero.

- **En Conclusión**

Queries de este tenor o de mayor complejidad, como por ejemplo buscar puntos fronteras de una imagen, u obtener pixels aislados (es decir pixels cuyos vecinos son de diferente color), zonas del mismo color, y otros podrían ser considerados.

Obsérvese que el query "Zonas de un mismo color" es posible. Si bien tal query es de Clausura Transitiva, y el mismo no es expresable en FO, ocurre que como la cardinalidad del dominio es fija, se puede conocer a priori un "diámetro" del grafo inducido por la relación Pixels-vecinos, y estimar de esta forma la cantidad de composiciones máxima que se necesitan para obtener el resultado.

CONCLUSIONES

Cuando se consideran Bases de Datos Relacionales cuya propiedad principal es contar con un dominio de cardinalidad fija, y tal cardinalidad es inalterable por cualquier transacción de actualización, es posible formular consultas a las mismas expresadas como Subfamilias Finitas de Circuitos Booleanos, puesto que constituyen un modelo teórico adecuado. Esto es porque tales subfamilias preservan la propiedad de Uniformidad al poder ser construidas homogénenamente por un algoritmo. También dada la adecuada relación entre tamaño y profundidad que mantienen los circuitos, se permite apreciar el grado de paralelización de la consulta, ya sea considerándoselo a nivel de circuito o a nivel de familia. Este formalismo sirve para mantener conjuntos de queries expresados como Subfamilias Finitas de Circuitos Booleanos, que pueden ser utilizados como primitivas implementadas a bajo nivel. También permite distinguir el poder expresivo del CP cuando del mismo se consideran subfamilias de fórmulas (enunciados) para la evaluación de una consulta, cuyo poder en estructuras relacionales finitas se hace equivalente a FO. Así las aplicaciones en donde el tamaño del dominio es fijo son aptas para mantener conjuntos de queries expresados como Subfamilias de Circuitos Booleanos.

Por ello, como futuras extensiones y aplicaciones de este trabajo, se pueden considerar casos como los expuestos, en donde los queries corrientes podrían definirse a priori y expresarse en el formalismo de los Circuitos Booleanos, con el fin de llevarlos a una implementación física, es decir a

nivel de hardware, con lo cual se obtendrían mejores tiempos de evaluación de queries, y además permitir la posibilidad de explotar el paralelismo propio del query.

Obviamente es necesario contar con bases de datos cuyo dominio tenga cardinalidad fija no modificable y con un conjunto de queries de interés corriente, lo cual constituye una limitación puesto que no en cualquier base de datos es posible hacer uso de este formalismo con real eficiencia. Este modelo es apropiado sólo para una subclase de bases, que son aquellas justamente que mantienen la cardinalidad fija como propiedad .

También, con este trabajo [PEREYRA 98], desde el punto de vista pedagógico, se contribuye a la construcción de una biblioteca de distintos tipos de formalismos, con diferentes grados de expresividad para formular consultas a Bases de Datos Relacionales. Actualmente se cuenta con evaluadores de expresiones algebraicas [GAGLIARDI 88], de expresiones de FO extendida [MALDOCENA 99], de un intérprete del lenguaje QL [BARROSO 97]; y de un traductor del Cálculo Relacional al Algebra Relacional con optimización de expresiones[YRUSTA 99]. Estos trabajos se encuentran disponibles como herramientas didácticas y de investigación.

REFERENCIAS BIBLIOGRÁFICAS

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

[ABITEBOUL 91] Abiteboul,S; Vianu, V.; "Generic Computation and Its Complexity", STOC 1991.

[BALCAZAR 88] Balcázar, J.L;Díaz, J; Gabarró, J; "Structural Complexity I". Springer- Verlag, 1988.

[BALCAZAR 90] Balcázar, J.L;Díaz, J; Gabarró, J; "Structural Complexity II". Springer- Verlag, 1990.

[BARROSO 97] Barroso, L.; Molina, G.; Quiroga, J.; "Implementación de un Interprete para el Lenguaje QL", Reporte Licenciatura en Cs. de la Computación, UNSL, 1997.

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

[COD 70] Codd, E.F.; "A relational model of data for a large shared data banks". Com of ACM 13/6):377-387,1970.

[DENENBERG 86] Denenberg, L.; Gurevich, Y; Shelah, S.; "Definability by Constant-Depth Polinomial- Size Circuits", Information and Control 70, 2/3 (reprint), 1986.

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

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

[GAGLIARDI 88] Gagliardi, E.; Quintas, S.; "Evaluador de Expresiones Algebraicas", Reporte Licenciatura en Cs. de la Computación, UNSL, 1998.

[HAMILTON 81] Hamilton, A.G.; "Lógica para matemáticos", Paraninfo, 1981.

[MALDOCENA 99] Maldocena,P.; Reyes,N; "Expresiones de Consultas a Bases de Datos mediante Extensiones de Lógica de Primer Orden", Reporte Licenciatura en Cs. de la Computación, UNSL, 1999.

[PEREYRA 98] Pereyra,S; Piffaretti,P. "Computación de queries utilizando circuitos lógicos". Reporte Licenciatura en Cs. de la Computación, UNSL, 1998.

[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, Jeffrey D. "Principles of Database and Knowledge Base Systems". Computers Science Press, 1988.

[YRUSTA 99] Yrusta,M. "Traducción del cálculo al álgebra con optimización". Reporte Licenciatura en Cs. de la Computación, UNSL, 1999.