Computación Paralela de Queries expresados con Circuitos Booleanos

Gagliardi, Edilma Olinda
Herrera, Norma Edith
Reyes, Nora Susana
Turull Torres, José María[1]

{oli, nherrera, nreyes, jmturull} @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, Argentina

ABSTRACT

Este trabajo se encuadra como una etapa de un proyecto mayor, en el que utilizamos los circuitos booleanos como un modelo teórico adecuado para la expresión de consultas a una base de datos relacional, estudiando diferentes aspectos de relevancia; en este caso, el grado de paralelización que poseen las mismas.

Para ello, consideramos la equivalencia entre lógica de primer orden y una clase restringida de familias de circuitos booleanos. Presentamos cómo transformar una consulta dada a otra equivalente, ambas expresadas en lógica de primer orden, de modo tal que traducida a una subfamilia finita de circuitos booleanos, resulte apropiada para el uso de recursos de paralelismo.

Analizamos la profundidad de los circuitos booleanos. Para ello trabajamos sobre los árboles de expresión de las fórmulas, buscando transformaciones sobre los mismos para lograr mínima profundidad en los circuitos booleanos asociados.

Palabras Claves

Teoría de la Computación, Bases de Datos Relacionales, Circuitos Booleanos, Computabilidad de Queries, Paralelismo, Lógica de Primer Orden.

[1] Universidad Tecnológica Nacional (FRBA);
Universidad Nacional de San Luis;
Subsidio de Universidad CAECE y Universidad Nacional de Luján

Introducción

Este trabajo forma parte de un proyecto mayor en el que se pretende proveer herramientas formales que permitan estudiar la computación de queries a bases de datos, dada la inadecuación a la Teoría de Computabilidad clásica. En la actualidad, la Informática ataca problemas de alta complejidad impulsada por un importante desarrollo tecnológico, haciendo más necesaria una teoría de base, sólida y dinámica, que permita utilizar herramientas formales. Estos desarrollos teóricos conducen a definir pautas de diseño en las bases de datos y en los queries que apunten a evitar los posibles conflictos que se desprenden de la inadecuación de los motores de bases de datos existentes a la teoría de computabilidad clásica.

Por ello, en el presente artículo mostramos cómo una consulta a una base de datos relacional puede ser expresada como una subfamilia finita de circuitos booleanos perteneciente a la clase AC^0, la cual puede evaluarse utilizando recursos de computación paralela.

En nuestro trabajo previo [10] mostramos la utilización de tales subfamilias finitas de circuitos booleanos como un modelo teórico adecuado para la expresión de consultas a una base de datos relacional, cuyo fundamento principal se basa en la equivalencia demostrada entre lógica de primer orden (FO) y una clase restringida de familias de circuitos booleanos. Este formalismo constituye un marco de estudio apropiado y tales subfamilias preservan la propiedad de Uniformidad, mostrando de qué manera son construibles a partir de un algoritmo que muestra por si mismo tal particularidad.

Se construye 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 $C=\{ C_0, C_1,\dots ,C_q \}$ con $q= n^r -1$, donde cada C_i decide si la i-esima r-tupla, considerándose un orden lexicográfico en las r-tuplas, pertenece o no al resultado del query. Así, para cada n y fijando la aridad del query en r, se construye una subfamilia finita de circuitos booleanos. La unión de todas las subfamilias, para cada n natural, constituye la familia infinita de circuitos que computa un query (función) dado.

Este problema visto en el marco del modelo de máquina de Turing, consistiría en tener una máquina cuya entrada es una fórmula y un número natural n, y cuya salida es una subfamilia de circuitos booleanos C, como se describió previamente.

En el presente trabajo buscamos una adecuada relación entre el tamaño y la profundidad de los circuitos, de forma tal que se permita apreciar el grado de paralelización de la consulta, ya sea considerándoselo a nivel de circuito o a nivel de la subfamilia de circuitos booleanos que la expresan. Al segundo aspecto lo dejaremos de lado, puesto que es trivial.

Nos interesa analizar a nivel de circuito porque es la representación de la estructura en sí de una consulta. La misma reemplazada por una expresión equivalente optimizada o transformada, nos permitirá obtener los circuitos booleanos de profundidad conveniente, para analizar el posible aprovechamiento de paralelismo que pueda contener efectivamente la fórmula.

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 de la consulta estará determinado por la profundidad del circuito. Así se verá que se podrá evaluar la fórmula en su máxima expresión de paralelismo.

En particular trabajamos con circuitos booleanos cuyo grado de entrada está limitado a una o dos entradas (bounded fan in) y cuyo grado salida es uno (bounded fan out), restringiéndonos a una clase particular de grafos dirigidos acíclicos (DAG's), en los cuales no se hace reuso de subcircuitos.

La estructura de los circuitos que representan la fórmula del query, es un árbol binario. En consecuencia, consideraremos los árboles de expresión de las fórmulas, haciendo las transformaciones que se requieran sobre los mismos, de manera tal que los árboles de los circuitos asociados, mantengan las propiedades que nos interesan para apreciar la máxima expresión de paralelismo.

En comienzo, presentamos los aspectos teóricos de interés para mostrar el marco apropiado de nuestra problemática y luego, brindamos las posibles transformaciones que permiten obtener consultas equivalentes a las originales y que son más apropiadas al caso que nos compete. Posteriormente, mostramos las subfamilias de circuitos booleanos correspondientes y analizamos cómo en algunos casos podemos optimizar los tiempos de computación usando paralelismo, cuando la génesis del problema lo permite, ya que no siempre es cierto que el tiempo necesario para la evaluación de la consulta disminuya sustancialmente por el sólo hecho de disponer de más de un procesador.

Circuitos Booleanos

Actualmente existe una variedad de modelos abstractos para computación paralela. Los circuitos booleanos constituyen uno de tales modelos. La complejidad de circuitos booleanos es de interés tanto en lo práctico como en lo teórico, y por ello mostraremos cómo en nuestro problema ellos constituyen un modelo apropiado para su estudio.

Los circuitos booleanos tienen como característica propia, a diferencia de otros modelos computacionales, que su entrada es de longitud fija por lo que para cada tamaño de entrada se debe construir el circuito booleano correspondiente. Esto conduce a la generación de una familia infinita de circuitos booleanos, uno por cada posible longitud de la entrada, y aún más, la estructura de los mismos puede llegar a variar según el tamaño de entrada dependiendo de la función que computen. Por ello, los circuitos booleanos son referidos como un modelo de computación no uniforme.

Si miramos una máquina de Turing o un lenguaje de programación, ellos pueden describir un algoritmo que resuelva un problema para cualquier entrada dada, independiente de la longitud de la misma. Tales modelos son referidos como modelos Uniformes de computación.

En nuestro caso, generamos una subfamilia finita de circuitos booleanos, dado que tanto la fórmula como la cardinalidad del dominio son considerados la entrada para el algoritmo que genera los mismos, por lo que se preserva la propiedad uniformidad.

Nuestros circuitos booleanos pertenecen a la clase NC, la cual es de fundamental importancia en la teoría de paralelismo, porque ella captura los problemas para los cuales se pueden describir algoritmos paralelos utilizando una cantidad factible de recursos de computación.

Se define

- Una *función booleana m-aria* como $f: \{0,1\}^m \rightarrow \{0,1\}$, para algún $m \in N$.
- Una *familia de funciones booleanas* como una secuencia $f= (f^n)_{n\in N}$ donde f^n es una función booleana n-aria.
- Una *base* como un conjunto finito de funciones booleanas y familias de funciones booleanas.

Obsérvese que una familia de funciones booleanas puede ser infinita mientras que una base es siempre finita.

Nosotros trabajaremos con la base $B = \{ \neg, \wedge^2, \vee^2 \}$, omitiendo familias de funciones booleanas dentro de la base.

Para nuestra problemática se define un *circuito booleano* 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 . Además, tienen grado de entrada limitado a una o dos entradas (bounded fan-in); y grado de salida limitado a uno (bounded fan-out). Obsérvese que, con estas restricciones, no hay rehuso de subcircuitos. Por lo que nuestra base es definida bounded fan-in limitado estándar.

Formalmente se puede definir un *circuito booleano* de la siguiente forma:

Sea B la base presentada. Un circuito booleano sobre B con dos entradas y una salida es una tupla C = (V, E, α, β, ω), donde (V, E) es un grafo dirigido acíclico, α: E $\rightarrow$N inyectiva, β: V$\rightarrow$ B $\cup$ {x_1, x_2 }, y ω:V $\rightarrow$ { y_1} $\cup$ {$*$} tal que las siguientes condiciones son mantenidas:

- Si v$\in$V tiene grado de entrada 0, entonces β(v) $\in$ {x_1, x_2} o β(v) es una función booleana 0-aria de B.
- Si v$\in$V tiene grado de entrada 0 < k < 2, entonces β(v) es una función booleana k-aria de B.
- Para i, $1 \leq i \leq 2$, existe a lo sumo un nodo v$\in$V, tal que $\beta(v) = x_i$.
- Existe un único nodo v$\in$V tal que $\omega(v)= y_1$.

Observar que x_1, x_2, y_1, $*$ son símbolos especiales asociados a ciertos nodos. Con β indicamos el tipo de compuerta. Con ω designamos cierto nodo como salida, por lo que con $\omega(v) = *$ designamos que no es nodo de salida. Con α establecemos un orden entre los arcos, lo cual induce un orden entre los nodos.

Así, un circuito booleano C computará una función booleana f : $\{0,1\}^2 \rightarrow \{0,1\}$. Este circuito booleano C=(V, E, α, β, ω) computa la función característica f_C de algún lenguaje A, por lo que diremos que C acepta A; luego si x $\in$ A* tal que $f_C(x) = 1$ entonces C acepta x.

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,… ,C_n, …}, donde cada C_n computa la función restringida al dominio $\{0,1\}^n$. Para nuestra problemática queda restringido a una o dos entradas y una única salida por compuerta.

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 la operación booleana de la compuerta; g_i el número de la compuerta, variable o constante, que provee la entrada izquierda a g , y g_d 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 $\overline{C}$.

Se define como tamaño de C, Size(C), a la cantidad de nodos o 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.

Se define para $i \geq 0$, NC^i = Size-Depth($n^{O(1)}$, $\log^i n$) y NC = $\cup_{i\geq 0}$ NC^i .

Si consideramos circuitos de tamaño polinomial y profundidad polilogarítmica, pero con (unbounded fan-in) grado de entrada no limitado, formalmente podemos ver las siguientes clases:

$$AC^i = \text{Unbounded Size-Depth } (n^{O(1)}, \log^i n), \text{ para } i \geq 0$$

$$AC = \cup_{i\geq 0} AC^i$$

La relación dada entre las clases de NC y AC es la siguiente: $NC^i \subseteq AC^i \subseteq NC^{i+1}$, para $i \geq 0$, de aquí que NC = AC.

Se define AC^0 = Unbounded Size-Depth ($n^{O(1)}$, $\log^0 n$)

Está demostrado que AC^0 es la clase de aquellos lenguajes que pueden ser definidos por fórmulas de primer orden, por lo que FO $\equiv AC^0$, [15], y $NC^0 \subseteq AC^0 \subseteq NC^1$, entonces cualquier fórmula de FO puede expresarse como una subfamilia de circuitos booleanos en la clase Size-Depth($n^{O(1)}$, log n).

Hemos observado que una familia de circuitos booleanos es infinita, sin embargo veremos que para nuestra problemática se construye una subfamilia de circuitos booleanos que aceptan una descripción finita y ello nos conduce a observar la uniformidad de la familia de los mismos.

Dado que consideramos sólo fórmulas en FO, y como FO tiene una caracterización en términos de complejidad de circuitos AC^0 uniforme, luego está en NC^1 uniforme, por lo que es expresable en circuitos limitados polinomialmente en tamaño.

En general, al transformar un DAG en árbol, podría obtenerse un circuito booleano exponencial en su tamaño. Pero, para obtener ese límite exponencial, el DAG original debería tener una profundidad lineal. En nuestra problemática, la expansión de un cuantificador de la fórmula en FO, es representada en el circuito booleano correspondiente por medio de un subárbol cuya profundidad está acotada logarítmicamente. Si consideramos que k es el rango cuantificacional de la fórmula y n la cardinalidad del dominio, entonces la profundidad final del árbol estará afectada en un factor k por la profundidad de los subárboles de expansión de los cuantificadores. Por consiguiente, en el peor caso, el ancho máximo del árbol será menor o igual a $2^{k \log n}$, con lo cual terminan siendo ancho y tamaño polinomial.

Nuestro objetivo consistirá en obtener los circuitos booleanos de tal subfamilia con profundidad logarítmica. De este modo, obtendremos una subfamilia perteneciente a la clase Size-Depth($n^{O(1)}$, log n), que constituyen la clase de funciones para las cuales es conveniente utilizar recursos de computación paralela.

La jerarquía NC representa una clase de complejidad de suma importancia para el desarrollo de algoritmos paralelos eficientes. En ella, la clase Size-Depth($n^{O(1)}$, log n) constituye la clase apropiada a nuestra problemática.

Referencias bibliográficas: [3,4], [7], [15].

Bases de Datos Relacionales y Queries

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 a la base de datos. Para el caso del modelo relacional, modelo de interés para el presente trabajo, Codd 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). Observando el modelo relacional en el marco de la teoría de modelos finitos, 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 [5] 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).

Sea σ es un vocabulario relacional finito, L_σ el lenguaje de primer orden con vocabulario σ, $\varphi \in L_\sigma$ con variables libres $\{x_1,x_2,...,x_r\}$, y B es una σ-estructura, con $d_1,\ldots,d_n \in D^B$. Denotaremos con la expresión $B \models \varphi (x_1, \ldots,x_r)[d_{s_1},...,d_{s_r}]$, 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 $1 \leq s_j \leq n$, $1 \leq j \leq r$. La acción de asignar el elemento d_{s_j} a la variable libre x_j, siendo x_j variable libre, es denominada valoración.

Nótese que de acuerdo a la semántica del lenguaje considerado, φ 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:

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

Referencias bibliográficas:[1], [5], [6], [8], [9], [11], [13], [14].

Traducción de FO a Circuitos Booleanos

Al interpretar una fórmula atómica se puede decir si ésta se satisface o no, es decir si es verdadera o falsa, en la interpretación para una valoración dada y se la puede ver como una proposición, asociándole una variable proposicional que la represente. De esta manera, las fórmulas atómicas expresadas en FO y valoradas, pueden ser traducidas al cálculo proposicional (CP).

Considérese 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 $1 \le h \le k$. Observaremos 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, tal que si pertenece a R_h^B, la proposición p tendrá el valor de verdad Verdadero, sino tendrá el valor de verdad Falso. 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.

Si $|D^B|^r = n^r$, entonces 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. La cantidad de variables proposicionales dependen de la aridad de la relación y la cardinalidad del dominio, y 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, numeradas desde el número $h = \Sigma_{j<i}\, n^{aj}$ hasta el número $(h+n^{(ai)}-1)$.

Siguiendo a [7][10][15] , dada una sentencia φ expresada en FO se puede construir el circuito C que representa a la fórmula de manera inductiva, construyendo un único nodo con arco (salida) 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.

Un aspecto a considerar es que para fórmulas del tipo $\varphi \equiv \forall x\ \varphi_1(x)$ sabemos que $\varphi_1(x)[d_1]$,…, $\varphi_1(x)[d_n]$ serán las fórmulas que deberán evaluarse y todas irán conectadas por el conectivo $\wedge$, por lo que el árbol de expresión se degeneraría en una lista. Por ello, en este caso, generaremos un árbol balanceado, (aplicando el algoritmo and-tree [13]), logrando la profundidad deseada. Con $\varphi \equiv \exists x\ \varphi_1(x)$ la construcción es similar, sólo que serán conectadas por el conectivo $\vee$. (en este caso usaremos el algoritmo or-tree).

Referencias bibliográficas:[7], [3,4], [10], [11], [12], [15].

Query expresado como Subfamilia finita de Circuitos Booleanos

Dada la base de datos $B=\langle D^B, R_1^B, \ldots, R_k^B \rangle$, con $D^B=\{d_1, d_2, \ldots, d_n\}$, $|D^B| = n$, la cantidad de posibles combinaciones de las r variables libres en D^B está definido por n^r. Así se nos presentan los siguientes casos:

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 la asignación del elemento k en la posición p de la tupla; 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 $C=\{ C_0, C_1, \ldots, C_q\}$, equivalente a n^r queries booleanos tal que para cada valoración $v_i : \{x_1, \ldots, x_r\} \rightarrow D^B$, el query Booleano φ_i es $\varphi(x_1, \ldots, x_r)[v_i(x_1), \ldots, v_i(x_r)]$, para $0 \leq i \leq q$.

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 $\varphi(x_1, \ldots, x_r)$.

Con respecto a la evaluación del query, denotamos con $B \models \varphi(x_1, \ldots, x_r)[d_{s_1}, \ldots, d_{s_r}]$, 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 $1 \leq s_j \leq n$, $1 \leq j \leq r$. Denotamos con φ^B, indistintamente, a la fórmula y a la relación que ella expresa:

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

representada en el formalismo de circuitos booleanos por un circuito C_h, para algún h dado, equivalente a φ_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

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

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

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

Referencias bibliográficas:[1], [9], [10], [12], [15].

Generación de Fórmulas Equivalentes Apropiadas para Paralelismo

La idea aquí es, dada una fórmula expresada en FO hallar otra equivalente en su semántica, tal que su árbol de expresión posea características especiales que nos lleven a obtener una subfamilia finita de circuitos booleanos cuya profundidad sea la mínima, y cuyos ancho y profundidad sean adecuados para enmarcarse en la clase NC^1, esto es Size-Depth($n^{O(1)}$, logn). Para ello, previamente introducimos los siguientes conceptos:

Un *árbol de expresión* para representar una fórmula φ de FO es un árbol binario, construido del siguiente modo:

- Si φ es una fórmula atómica $R(y_1, \ldots, y_r)$, entonces se representa mediante un nodo con sus subárboles correspondientes vacíos y conteniendo, como un atributo del nodo, al conjunto de las variables libres.
- En otro caso, siendo φ_1 y φ_2 fórmulas, si ocurre que:
  a) $\varphi = \varphi_1 \wedge \varphi_2$ o $\varphi = \varphi_1 \vee \varphi_2$, entonces φ se representa con un árbol de expresión cuya raíz es un nodo rotulado con el conectivo respectivo y cuyos subárboles izquierdo y derecho corresponden a los árboles de expresión de φ_1 y φ_2 respectivamente.
  b) $\varphi = \neg\varphi_1$, entonces φ se representa con un árbol de expresión cuya raíz es un nodo rotulado con el conectivo ¬ y cuyo subárbol izquierdo corresponde al árbol de expresión de φ_1 .
  c) $\varphi = \forall x\ \varphi_1$ o $\varphi = \exists x\ \varphi_1$, entonces φ se representa con un árbol de expresión cuya raíz es un nodo rotulado con el respectivo cuantificador, cuyo subárbol izquierdo corresponde al árbol de expresión de φ_1 y que contiene, como un atributo del nodo, la variable x.

Definimos *peso* de un nodo a una función total $P : V \rightarrow N$ (Naturales), donde V es el conjunto de nodos del árbol de expresión de una fórmula, y donde P aplicada a un nodo *v* de V, indica la profundidad del subcircuito booleano correspondiente que se generará a partir de *v*.

Sea *v* un nodo del árbol de expresión de una fórmula de FO, definimos recursivamente la función P del siguiente modo:

- Si *v* representa a una fórmula atómica $R(y_1, \ldots, y_r)$, entonces $P(v) = 1$.
- Si *v* representa a una fórmula $\varphi_1 \wedge \varphi_2$ o $\varphi_1 \vee \varphi_2$, y siendo v_1 y v_2 sus hijos izquierdo y derecho respectivamente, es decir son las raíces de los subárboles que representan a φ_1 y a φ_2, entonces $P(v) = 1 + máx.(P(v_1), P(v_2))$.
- Si *v* representa a una fórmula $\neg\varphi_1$ y si v_1 es su hijo izquierdo, entonces $P(v) = 1 + P(v_1)$.
- Si *v* representa a una fórmula $\forall x\ \varphi_1$ o $\exists x\ \varphi_1$, donde v_1 es su hijo izquierdo, entonces $P(v) = 1 + \lceil \log n \rceil + P(v_1)$, donde n es la cardinalidad del dominio de la estructura.

Definimos *ancho* de un nodo a una función total $A : V \rightarrow N$, donde V es el conjunto de nodos del árbol de expresión de una fórmula, y donde A aplicada a un nodo *v* de V, indica la cantidad de hojas del subcircuito booleano correspondiente que se generará a partir de *v*.

Sea *v* un nodo del árbol de expresión de una fórmula de FO, definimos recursivamente la función A del siguiente modo:

- Si *v* representa a una fórmula atómica $R(y_1, \ldots, y_r)$, entonces $A(v) = 1$.
- Si *v* representa a una fórmula $\varphi_1 \wedge \varphi_2$ o $\varphi_1 \vee \varphi_2$, y siendo v_1 y v_2 sus hijos izquierdo y derecho respectivamente, es decir son las raíces de los subárboles que representan a φ_1 y a φ_2, entonces $A(v) = A(v_1) + A(v_2)$.
- Si *v* representa a una fórmula $\neg\varphi_1$ y si v_1 es su hijo izquierdo, entonces $A(v) = A(v_1)$.
- Si *v* representa a una fórmula $\forall x\ \varphi_1$ o $\exists x\ \varphi_1$, donde v_1 es su hijo izquierdo, entonces $A(v) = n \times A(v_1)$.

De este modo, al hacer el análisis sintáctico de la fórmula expresada en FO, por cada nodo correspondiente a un conectivo lógico o a un cuantificador, se considerará la estructura del circuito booleano que generará tal nodo, y en consecuencia recibirá el peso y el ancho apropiado. Por ejemplo, para los cuantificadores universal y existencial, sabemos que en la creación de los circuitos booleanos se reemplazarán por un árbol balanceado (mediante el uso de los algoritmos and-tree y or-tree), y por ello debe considerarse a ese nodo con el peso y ancho relativo a lo que potencialmente generará. Así, en el caso de cuantificar una fórmula atómica, esto es *log n* y *n* respectivamente, siendo *n* la cardinalidad del dominio.

Como caso particular podemos considerar aquel árbol en el cual exista algún nodo que es raíz de un subárbol " no balanceado". Esto es porque los pesos de sus hijos difieren en más de uno, y no hay forma de hacer cambios para equiparar los mismos, puesto que de hacerlo se obtendría una fórmula no equivalente semánticamente a la original (la expresión obtenida no representaría el mismo query). Por ejemplo, esto ocurre cuando hay algún cuantificador inmerso en alguno de los subárboles, $(R(x) \vee \forall y\varphi_1)$ (o análogamente $(R(x) \wedge \exists y\varphi_1)$), donde el nodo correspondiente a la primera subfórmula tiene peso 1 y el de la otra peso mayor que log n.

Cabe notar que hemos usado informalmente la expresión árbol "balanceado", sin especificar su definición. En realidad, abandonaremos esta expresión reemplazándola por la de árbol "parcialmente balanceado", puesto que no se corresponde con la visión de balanceo habitualmente utilizada.

Dada φ una fórmula de FO, mostramos a continuación un método que permite obtener un *árbol parcialmente balanceado en peso P de* φ.

Sea T un árbol de expresión que representa una fórmula φ de FO, entonces en él se pueden distinguir distintos tipos de subárboles T'. Cada T´ puede ser:

- el árbol de expresión de una *fórmula atómica.*
- el árbol de expresión consistente de un único nodo representando un *cuantificador* ($\forall$,$\exists$).
- el árbol de expresión consistente de un único nodo representando el conectivo *negación* ($\neg$).
- el árbol de expresión de una fórmula, el cual es maximal en cantidad de nodos y en el que todos los nodos tienen el mismo conectivo *and* ($\wedge$) u *or* ($\vee$).

Entonces, para cada subárbol T´ de T, si pertenece al primer caso se lo considera trivialmente como árbol parcialmente balanceado.

Para los casos restantes se debe verificar si todos los subárboles que se desprenden de tal árbol han sido tratados y son árboles parcialmente balanceados porque pudieron haber modificado sus pesos. Si ello ocurre, considerando T´, aparecen los siguientes casos:

- T´ es un nodo con cuantificador o con conectivo negación: se actualiza su peso en función del nuevo peso de su subárbol , el cual ya es parcialmente balanceado.
- En otro caso (esto es con conectivos $\wedge$ u $\vee$) se aplica *método de balanceo.*

El *método de balanceo* consiste en tomar todos los pesos de los subárboles que se desprenden de T´ y cuyas raíces no son nodos de T´. Sean φ_1, φ_2,…, φ_k las fórmulas que representan dichos subárboles y sean p_1, p_2,…,p_k los pesos asociados a las raíces de tales subárboles. Se ordenan los pesos en forma creciente, obteniendo una secuencia $s = <p_{i1}, p_{i2}, ..,p_{ik}>$, con $1 \leq i \leq k$. Entonces se analiza:

- Si todos los pesos tomados de a pares difieren en más de uno, se toma la secuencia ordenada y comenzado desde el menor hacia el mayor, se arma un nuevo árbol de expresión que presupone el mejor balanceo.

  Supongamos que el conectivo considerado sea $\wedge$, este nuevo de árbol de expresión representará la fórmula $(...(\varphi_{i1} \wedge \varphi_{i2}) \wedge \varphi_{i3}) \wedge\ldots)\wedge\varphi_{ik})$, que es equivalente a la fórmula original,

dadas las propiedades de asociatividad y conmutatitividad de dicho conectivo. Lo mismo se aplicará al conectivo or ($\vee$).

- En otro caso, si existe una subsecuencia maximal $s' = <p_i,..., p_j>$ *de* s, tal que entre cualquier par de pesos hay una diferencia de a lo sumo uno, se arma un árbol balanceado usando el algoritmo and-tree u or-tree, dependiendo del caso, y se reemplaza la subsecuencia s' en s por el peso de la raíz del nuevo árbol. Luego, se aplica el método de balanceo a la nueva secuencia.

Estas modificaciones en el árbol de expresión pueden ser realizadas dado que los conectivos que estamos considerando , and y or, satisfacen las propiedades conmutativa y asociativa.

De esta manera se obtiene un árbol de expresión equivalente en su semántica a la expresión original, y parcialmente balanceado en peso, el cual será más adecuado a la computación paralela por su mejor profundidad.

Observemos que el ancho de un árbol de expresión de una fórmula de FO define la cantidad de procesadores necesarios para procesar la misma en su máxima expresión de paralelismo. En consecuencia, con A(v) indicamos la cantidad de procesadores requeridos para la fórmula representada por el árbol de expresión con raíz v, bajo estas condiciones.

Dada φ una fórmula de FO, con árbol de expresión T parcialmente balanceado tenemos que:

- si T es totalmente balanceado (en su concepción habitual), podemos concluir que la fórmula es bien paralelizable y si r es la raíz de T, tan sólo son necesarios A(r) procesadores.
- en otro caso, tenemos que existe al menos un nodo v de T que es raíz de un árbol donde los pesos de los subárboles difieren en más de uno. Sean T_1 y T_2 los subárboles izquierdo y derecho de v, cuyas raíces son v_1 y v_2 respectivamente, tal que $P(v_1) >> P(v_2)$ (el caso simétrico es análogo). Denotamos con D a la diferencia de niveles existentes entre ambos subárboles, según el circuito booleano que representa v; esto es $D = P(v_1) - P(v_2)$. Sean $A(v_1)$ y $A(v_2)$ los anchos respectivos de v_1 y v_2. Sea T_1' la restricción del árbol T_1 al nivel $(P(v_1) – D)$, o equivalentemente al nivel $P(v_2)$, y sea h la cantidad de hojas que posee T_1'. La variable h representa la cantidad real de procesadores que aún son necesarios y utilizados para el procesamiento del nivel $(P(v_1) – D)$ del árbol T_1. La cantidad de procesadores liberados en tal nivel puede calcularse por medio de la expresión $(A(v_1) – h)$, que denotaremos con *l*. Entonces, teniendo la cantidad de procesadores en uso, h, y la cantidad de procesadores libres, *l*, podemos analizar la relación existente entre la cantidad de procesadores liberados y la cantidad necesaria para iniciar el procesamiento del subárbol derecho de v: $A(v_2)$. De esta forma se dan dos situaciones posibles:
  a) $l \geq A(v_2)$: podemos iniciar procesamiento paralelo en el subárbol derecho de v.
  b) $l < A(v_2)$: no es posible iniciar el procesamiento paralelo en su máxima expresión de paralelismo.

  Notemos que si llegamos a la raíz del árbol mediante la primera situación, obtenemos un resultado semejante a tener un árbol totalmente balanceado. Es decir, que todos los niveles del árbol T fueron barridos paralelamente y nunca ocurrió una situación de corte de paralelismo que provocase inevitablemente un procesamiento secuencial.

Referencias bibliográficas:[3,4], [10], [12].

Ejemplo

Sea φ de FO la siguiente fórmula:

$$\varphi: (((U(x_1, x_2, x_3) \wedge T(x_1, x_2, x_3)) \wedge \exists x_2 (R(x_1, x_2, x_3) \vee E(x_1, x_2, x_3)) \wedge S(x_1, x_2, x_3)) \wedge \neg R(x_1, x_2, x_3)$$

En la Figura 1.a) mostramos gráficamente el árbol de expresión que representa a φ. La Figura 1.b) ilustra los distintos tipos de subárboles de expresión que distinguimos para nuestro análisis. El único subárbol que no se encuentra parcialmente balanceado es aquel al que nombramos T'. La

Figura 1.c) indica los subárboles que se desprenden de T´, y que representan las fórmulas φ_1, φ_2, …, φ_5, indicando los pesos correspondientes a cada uno de ellos. Habiendo aplicado el método de balanceo, se obtiene el árbol de expresión de una fórmula φ', equivalente a φ, el cual es parcialmente balanceado, donde φ' es:

$$\varphi': ((((U(x_1, x_2, x_3) \wedge T(x_1, x_2, x_3)) \wedge S(x_1, x_2, x_3)) \wedge \neg R(x_1, x_2, x_3)) \wedge \exists x_2 (R(x_1, x_2, x_3) \vee E(x_1, x_2, x_3)))$$

La Figura 1.d) ilustra el árbol de expresión de la fórmula φ' y cuyo árbol se encuentra parcialmente balanceado. Se puede observar que la profundidad del árbol que representa a φ' es menor que la profundidad del que representa a φ.

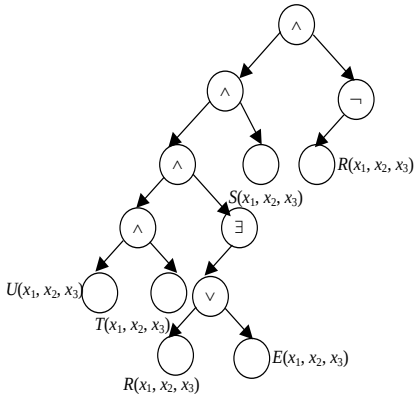

Figura 1.a) Árbol de expresión de la fórmula φ

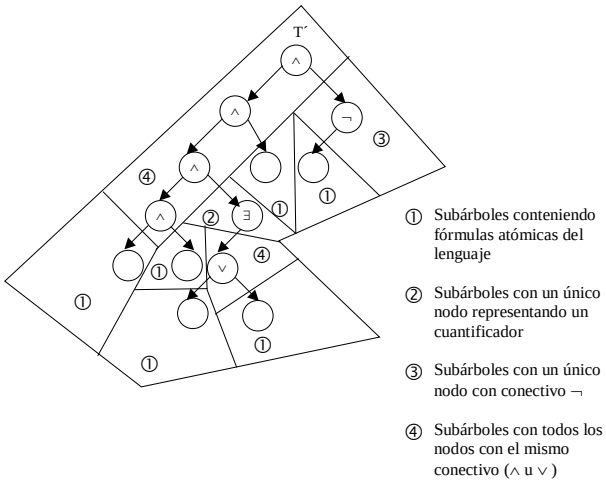

Figura 1.b) Distinción de los distintos tipos de subárboles presentes en el árbol de Figura 1.a)

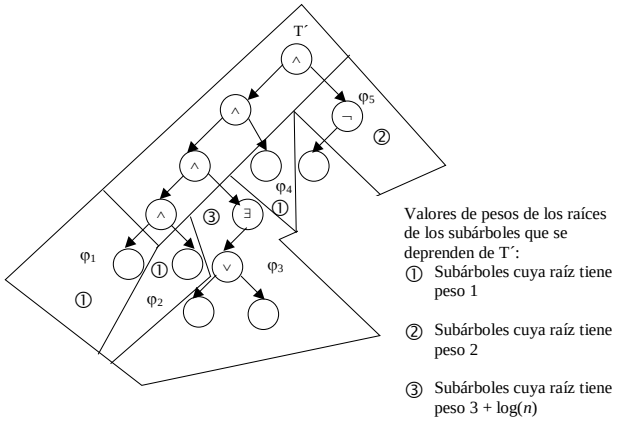

Figura 1.c) T´ junto a los pesos de los subárboles que se desprenden de él

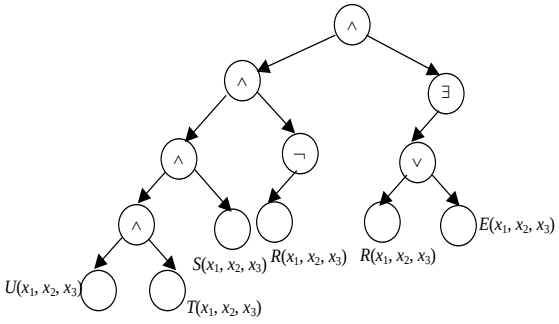

Figura 1.d) Árbol de expresión parcialmente balanceado que representa a φ'

Conclusiones

Como se mencionó al principio, este trabajo forma parte de un proyecto mayor en el que se pretende consolidar y alimentar una línea de estudio, con el fin de proveer herramientas formales que permitan estudiar la computación de queries a bases de datos, dada la inadecuación a la Teoría de Computabilidad clásica con respecto a este tópico.

El punto esencial es que en la medida en que la Informática pretende atacar problemas de alta complejidad, lo que se ha tornado habitual debido al tremendo desarrollo en cuanto a tecnología informática, se hace más necesaria una teoría de base, sólida y dinámica, que permita utilizar herramientas formales para enfrentar esos problemas. De allí que pensemos que es altamente

relevante hoy en día el desarrollo de las teorías de base que permitan comprender cabalmente los fenómenos que subyacen en las distintas áreas de las Ciencias de la Computación, fenómenos que utilizamos, pero que no alcanzamos a comprender plenamente.

Por otra parte, estos desarrollos teóricos, nos permitirán definir pautas de diseño en las bases de datos que apunten a evitar los posibles conflictos que se desprenden de la inadecuación de los motores de bases de datos existentes a la teoría, antes mencionada.

Los circuitos booleanos conforman un modelo teórico adecuado para el estudio de las consultas a bases de datos relacionales. A partir de un query de una cierta aridad y un número natural n que representa la cardinalidad del dominio de la base de datos, mostramos cómo obtener una subfamilia finita de circuitos booleanos que preservan la propiedad de uniformidad, mantienen una adecuada relación entre tamaño y profundidad, y que permiten apreciar el grado de paralelización de la consulta. Para ello se trabaja sobre la expresión lógica, reemplazándola por otra equivalente en su semántica, buscando que la estructura de los circuitos pertenecientes a la subfamilia que representa la consulta, sea la más apropiada en relación a su mínima profundidad.

Como futuros trabajos quedan pendientes generalizar resultados a DAG's sin restricciones sobre los grados de entrada y/o salida, continuar con la problemática de optimización de queries, observando la computación paralela y la secuencial, y como un correlato la computación de queries en bases de datos distribuidas. También dada la pobre expresividad de FO, se pretende llevar estos desarrollos teóricos a lógicas con mayor poder expresivo, ya sea extendiendo FO con otros cuantificadores o tratando subconjuntos de lógicas de orden superior.

Referencias Bibliográficas

1 Abiteboul,S; Hull and Vianu, V.; "Foundations of Databases". Addison-Wesley Publishing Company, 1995.

2 Abiteboul,S; Vianu, V.; "Generic Computation and Its Complexity", STOC 1991.

3 Balcázar, J.L;Díaz, J; Gabarró, J; "Structural Complexity I". Springer- Verlag, 1988.

4 Balcázar, J.L;Díaz, J; Gabarró, J; "Structural Complexity II". Springer- Verlag, 1990.

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

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

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

8 Ebbinghaus, H; Flum, J.; "Finite Model Theory" , Springer-Verlag, 1995.

9 Ebbinghaus, H; Flum, J.;Thomas, W.; "Mathematical Logic", Springer-Verlag, 1984.

10 Gagliardi, E.O.; Grosso, A.L; et al, "Computación de queries a bases de datos relacionales utilizando circuitos Booleanos", CACIC'99, Tandil, Argentina.

11 Hamilton, A.G.; "Lógica para matemáticos", Paraninfo, 1981.

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

13 Turull Torres, J.M. "Clases de Bases de Datos L-Rígidas y Expresividad de Lenguajes Relacionales Incompletos". Tesis Doctoral, UNSL, 1996.

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

15 Vollmer, Heribert, "Introduction to Circuit Complexity, a uniform approach". Springer Verlag, 1999.