Foundations of Query Languages

Dr. Fang Wei

Lehrstuhl für Datenbanken und Informationssysteme
Universität Freiburg

SS 2011

Measurement of Query Processing

- Complexity classes are usually defined for Decision (yes/no) problems.
- Queries may have a large output.
- It would be unfair to count the size of the output as *complexity*.
- We therefore consider the following decision problems, which areall computationally equivalent purposes (logspace equivalent).
  - Boolean Queries $D \models Q$?
  - The Query of Tuple problem (QOT)$t \in Q(D)$?
  - The Query-Emptiness problem $Q(D) \neq \emptyset$

Measurement of Query Processing

- Data Complexity $Q(D)$: where D is considered as input, while Q as fixed.
- Query Complexity $Q(D)$: where Q is considered as input, while D as fixed.
- Combined Complexity $Q(D)$: both D and Q are considered as input.

QBF Problem

QBF (Quantified Boolean Formulas) (a.k.a. QSAT)

$$Q_1 x_1 Q_2 x_2 \dots Q_n x_n \phi(x_1, x_2, \dots, x_n)$$

$Q_i \in \{\exists, \forall\}$ is a quantifier, ϕ is a Boolean formula in CNF; $Q_1 \dots Q_n$ alternates between $\exists$ and $\forall$.
For instance, $\exists x_1 \forall x_2 \dots \exists x_n$

QBF Example

$$\exists x_1 \forall x_2 \exists x_3 (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2)$$

$$\exists x_1 \forall x_2 \exists x_3 (x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$

QBF Problem - Algorithm

Algo *Truth* (Φ)

if Φ is quantifier-free **then** return its truth value

end if

Let $\Phi = Q_1 x_1 \ldots Q_n x_n \, \phi(x_1, \ldots, x_n)$;

$b_0 = \mathit{Truth}(Q_2 x_2 \ldots Q_n x_n \, \phi(0, x_2, \ldots, x_n))$;

$b_1 = \mathit{Truth}(Q_2 x_2 \ldots Q_n x_n \, \phi(1, x_2, \ldots, x_n))$; % re-using space

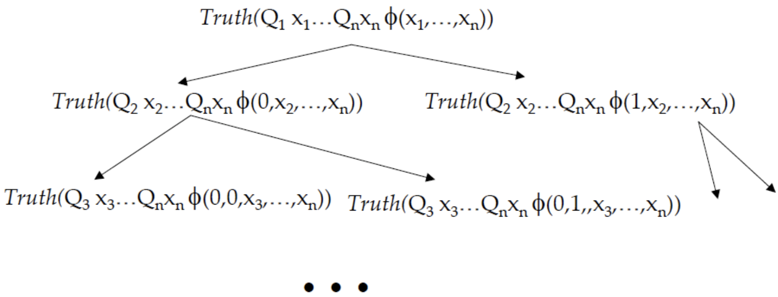
if $Q_1 = \exists$ **then** return $b_0 \vee b_1$

else return $b_0 \wedge b_1$

end if

end Algo

QBF Problem - Algorithm



Depth of recursion: n, at each step the stack size is $poly(n)$
$\rightarrow$ QBF in PSPACE

Logspace Transducer

A logspace transducer is a Turing machine with a read-only input tape, a write-only output tape, and a read/write work tape.

- The work tape may contain $O(\log n)$ symbols.
- A logspace transducer M computes a function $f : \Sigma^* \longrightarrow \Sigma*$, where $f(w)$ is the string remaining on the output tape after M halts when it is started with w on its input tape.
- f is a logspace computable function

Logspace computation

Some capabilities of logspace machines

- Maintain (a constant number) of counters on the worktape, and increment or decrement such counter
- Locate particular items of a well-structured input (integers, list item) via pointers consisting of input-tape address
- Access, process and compare input items in a bit-by-bit fashion

Binary integer addition, subtraction, comparing data items or lists, copying data items, searching, sorting, etc.

Language in Logspace

$$A = \{0^k 1^k | k \geq 0\}$$

is a member of Logspace.

- on the work tape, maintain a counter C, and one pointer P to the input tape
- Read the content of what P is pointing to; increase the counter as long as the content is 0 (move P to right)
- As soon as the first 1 is met, start decreasing the counter
- Accept if read the end and the counter is set to 0, reject otherwise

Join Operation in Logspace

Join of two relations r and s on attribute A

- Maintain two tuple pointers τ_r and τ_s
- Outer loop: move τ_r to point successively to each tuple of r
- For each such tuple t_r, an inner loop makes τ_s successively point to each tuple t_s
- For each combination of tuples t_r and t_s, identify the fields corresponding to attribute A and check, bit by bit, where the A-value of t_r and t_s are equal
- If yes, the relevant part of t_r and t_s are (bitwise) copied to the output tape
- Space used on work tape: 4 pointers + 2 + a constant number of counters

Complexity of FO Queries

Theorem

Evaluating Boolean FO (or RA) queries is as follows:

- *PSPACE complete (combined complexity)*
- *PSPACE complete (query complexity)*
- *in LOGSPACE (data complexity)*

The same complexity results apply to the Query-emptiness problem and to the QOT-Problem

FO Evaluation

Eval(I, φ)
Case
  φ is $p(t_1,\ldots,t_k)$: return $p^I(t_1,\ldots,t_k)$
  φ is $\neg\,\psi$: return $\neg$ Eval(I,ψ)
  φ is $\theta \wedge \psi$: return Eval(I,θ) $\wedge$ Eval(I,ψ)
  φ is $(\exists x)\psi$:
    B:= false
    for $a \in$ **dom** do
      B := B $\vee$ Eval(I, $\varphi_{x \to a}$)
    return B
End Case

FO Complexity

$$m(\log m + m \log n)$$

- Combined complexity (both m and n as input):
  $m(\log m + m \log n) \rightarrow$ PSPACE
- Data complexity (n as input, m as constant):
  $\log n \rightarrow$ LOGSPACE
- Query complexity: (m as input, n as constant) :
  $m^2 \rightarrow$ PSPACE

FO Complexity: Combined Complexity

PSPACE hardness proof: polynomial reduction from QBF problem (PSPACE complete)

- dom: $\{1, 0\}$
- Database: true(1), false(0)
- $\exists x_1 \forall x_2 \exists x_3 (x_1 \vee \neg x_2 \vee x_3) \wedge \ldots$
  $\exists x_1 \forall x_2 \exists x_3 (\mathit{true}(x_1) \vee \neg \mathit{false}(x_2) \vee \mathit{true}(x_3)) \wedge \ldots$

Data Complexity

Data complexity (n as input, m as constant): $\log n \longrightarrow$ in Logspace
Can we get a better upper bound?

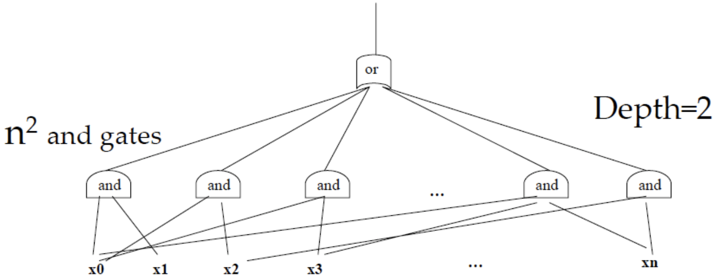
$$AC0 \subset NC1 \subseteq L \subseteq NL \subseteq LOGCFL \subseteq NC2$$

We first need to define circuit complexity classes.

Boolean Circuits

- size= # gates
- depth= longest path from input to output
- formula(or expression): graph is a tree
- can build circuit family that decides L

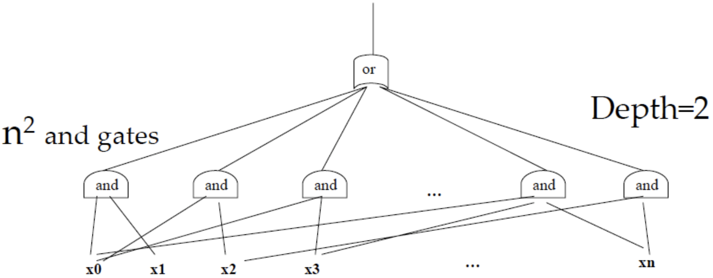
Circuit Family



What is the function of this circuit?

It accepts the language that consists of strings with at least two 1's.

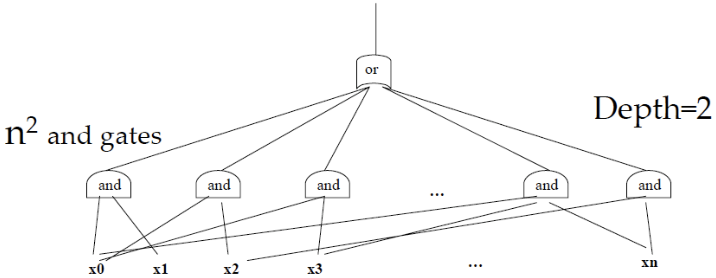
Circuit Family



What is the function of this circuit?

It accepts the language that consists of strings with at least two 1's.

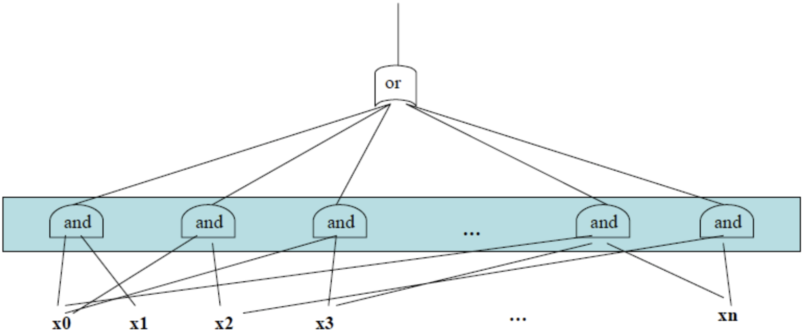
Circuit Family



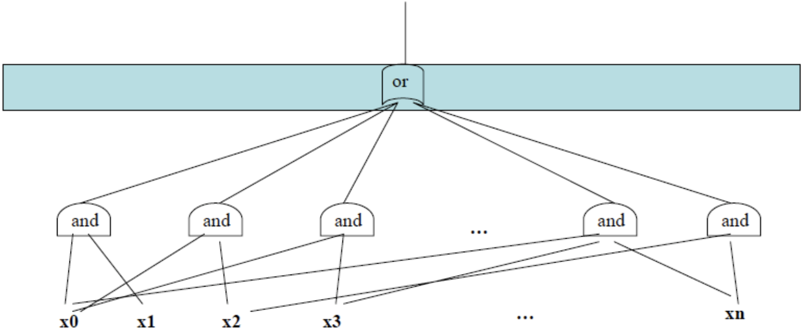
What is the function of this circuit?

It accepts the language that consists of strings with at least two 1's.

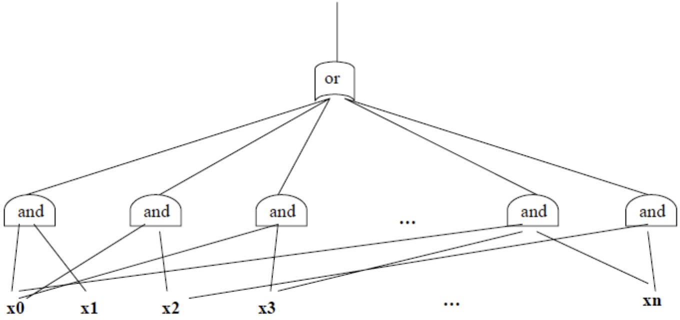
Circuit Family



Circuit Family



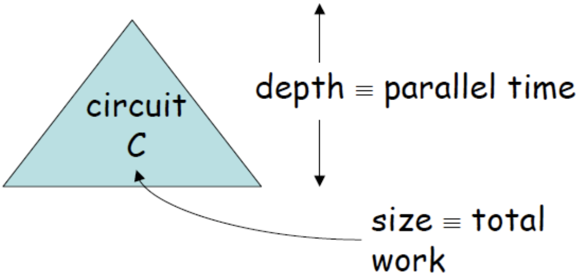
Circuit Family



2 steps, $O(n)$ processors

Parallelism

uniform circuits allow refinement of polynomial time:



Small Depth Circuit

A small depth circuit is a polynomial-size circuit whose depth is poly-logarithmic in its size. That is: a circuit with
size=poly(n) and depth=$O(\log^k n)$
Such circuits capture the notion of efficient parallel computation.

NC and AC

- $NC = \bigcup_k NC^k$ (Nick's Class) class of languages decided by families of fan-in 2 circuits $\{C_n\}$ s.t.
  size(C_n)=poly(n) and depth(C_n)=$O(\log^k n)$.
- $AC = \bigcup_k AC^k$, defined analogously, difference: arbitrary fan-in allowed.

Examples $\Sigma = \{0, 1\}$

- $1 \cdot \Sigma^* \cdot 1 \in NC0$
- $\{0^k 1^k (k \geq 0)\} \in AC0$
- PARITY $:= \{w \in \Sigma^* | \#1(w) odd\} \in NC1$

Data Complexity

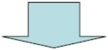
Theorem

Data complexity of FO query: Complete for logtime uniform AC0.

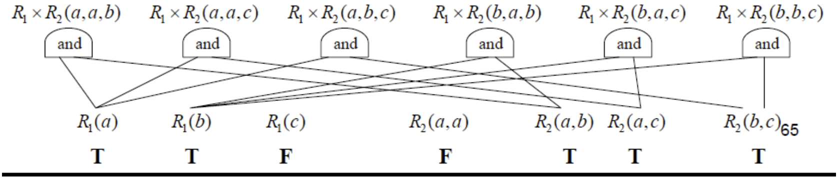
$$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1$$

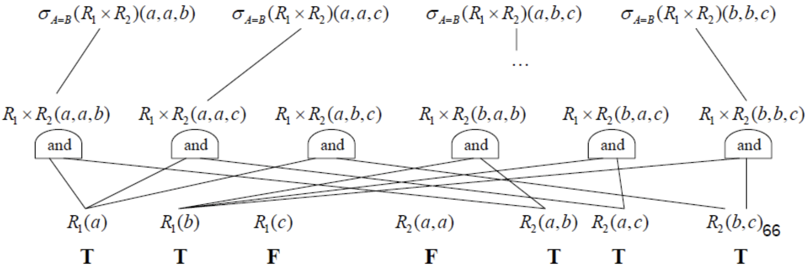
R_1	A
	a
	b

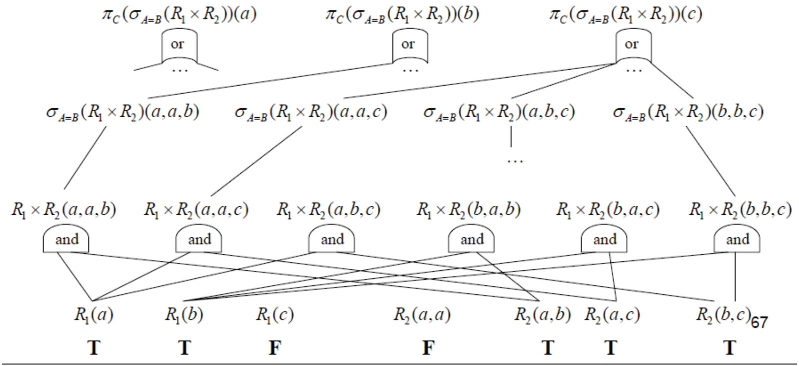
R_2	B	C
	a	b
	a	c
	b	c

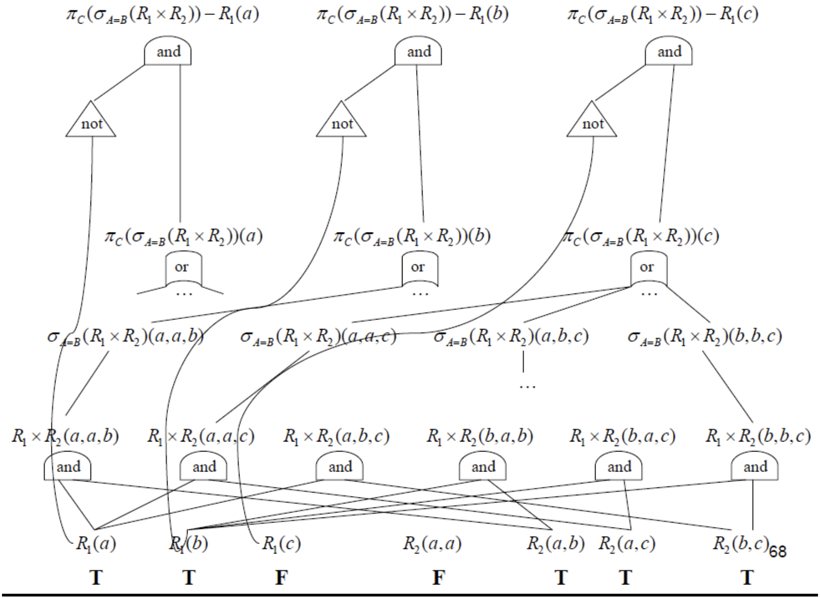


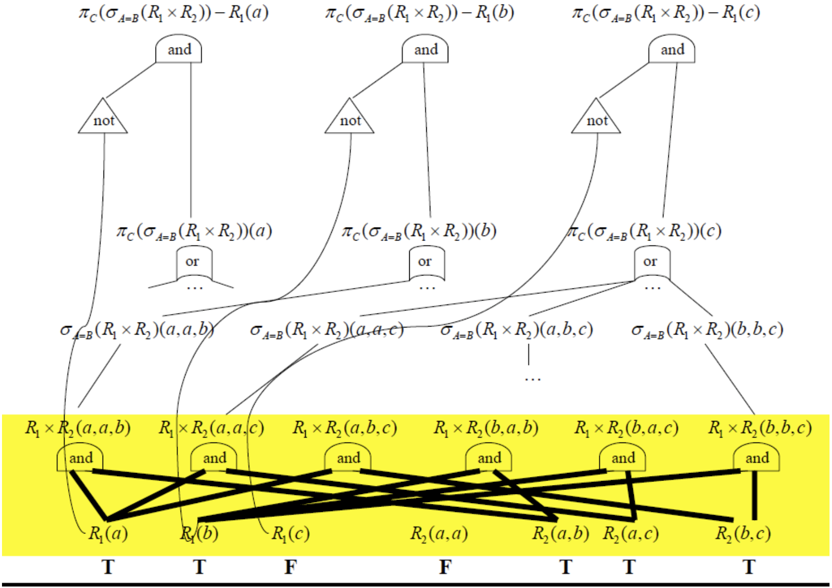
$R_1(a)$	$R_1(b)$	$R_1(c)$	$R_2(a,a)$	$R_2(a,b)$	$R_2(a,c)$	$R_2(b,c)$
T	**T**	**F**	**F**	**T**	**T**	**T**

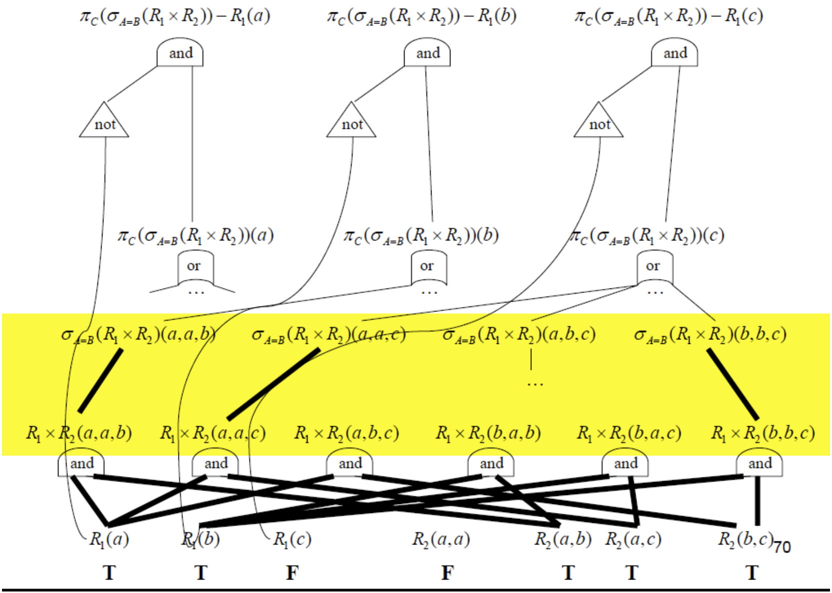

$R_1 \times R_2(a,a,b)$
$R_1 \times R_2(a,a,c)$
$R_1 \times R_2(a,b,c)$
$R_1 \times R_2(b,a,b)$
$R_1 \times R_2(b,a,c)$
$R_1 \times R_2(b,b,c)$
and
and
and
and
and
and
$R_1(a)$
$R_1(b)$
$R_1(c)$
$R_2(a,a)$
$R_2(a,b)$
$R_2(a,c)$
$R_2(b,c)$
65
T
T
F
F
T
T
T

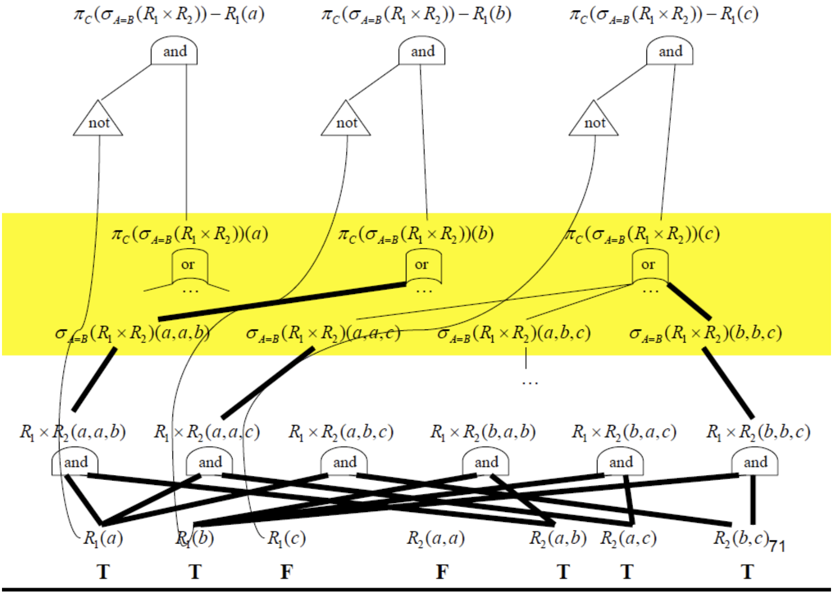

$\sigma_{A=B}(R_1 \times R_2)(a,a,b)$
$\sigma_{A=B}(R_1 \times R_2)(a,a,c)$
$\sigma_{A=B}(R_1 \times R_2)(a,b,c)$
$\sigma_{A=B}(R_1 \times R_2)(b,b,c)$
...
$R_1 \times R_2(a,a,b)$
$R_1 \times R_2(a,a,c)$
$R_1 \times R_2(a,b,c)$
$R_1 \times R_2(b,a,b)$
$R_1 \times R_2(b,a,c)$
$R_1 \times R_2(b,b,c)$
and
and
and
and
and
and
$R_1(a)$
$R_1(b)$
$R_1(c)$
$R_2(a,a)$
$R_2(a,b)$
$R_2(a,c)$
$R_2(b,c)$
66
T
T
F
F
T
T
T


$\pi_C(\sigma_{A=B}(R_1 \times R_2))(a)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(b)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(c)$
or
or
or
$\sigma_{A=B}(R_1 \times R_2)(a,a,b)$
$\sigma_{A=B}(R_1 \times R_2)(a,a,c)$
$\sigma_{A=B}(R_1 \times R_2)(a,b,c)$
$\sigma_{A=B}(R_1 \times R_2)(b,b,c)$
$R_1 \times R_2(a,a,b)$
$R_1 \times R_2(a,a,c)$
$R_1 \times R_2(a,b,c)$
$R_1 \times R_2(b,a,b)$
$R_1 \times R_2(b,a,c)$
$R_1 \times R_2(b,b,c)$
and
and
and
and
and
and
$R_1(a)$
$R_1(b)$
$R_1(c)$
$R_2(a,a)$
$R_2(a,b)$
$R_2(a,c)$
$R_2(b,c)$
67
T
T
F
F
T
T
T

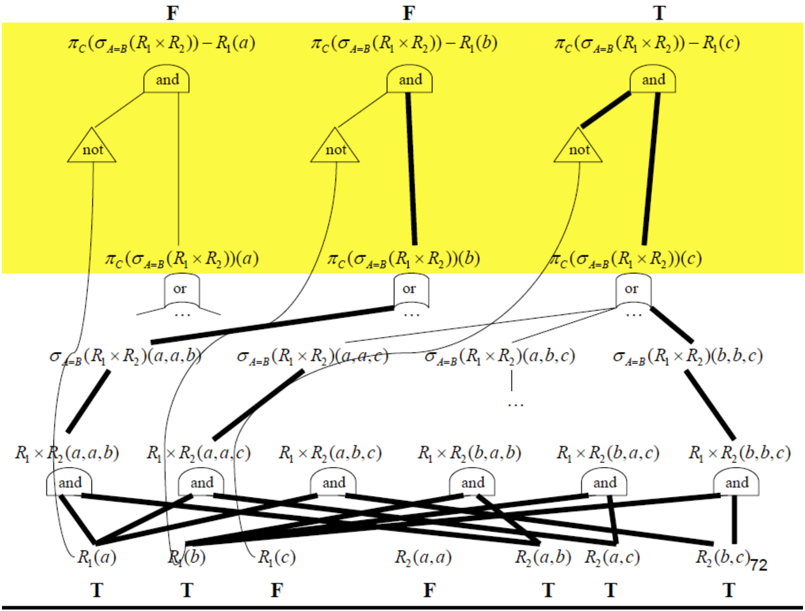

$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(a)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(b)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(c)$
and
not
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(a)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(b)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(c)$
or
...
$\sigma_{A=B}(R_1 \times R_2)(a,a,b)$
$\sigma_{A=B}(R_1 \times R_2)(a,a,c)$
$\sigma_{A=B}(R_1 \times R_2)(a,b,c)$
$\sigma_{A=B}(R_1 \times R_2)(b,b,c)$
$R_1 \times R_2(a,a,b)$
$R_1 \times R_2(a,a,c)$
$R_1 \times R_2(a,b,c)$
$R_1 \times R_2(b,a,b)$
$R_1 \times R_2(b,a,c)$
$R_1 \times R_2(b,b,c)$
$R_1(a)$ T
$R_1(b)$ T
$R_1(c)$ F
$R_2(a,a)$ F
$R_2(a,b)$ T
$R_2(a,c)$ T
$R_2(b,c)$ T
68


$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(a)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(b)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(c)$
and
not
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(a)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(b)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(c)$
or
...
$\sigma_{A=B}(R_1 \times R_2)(a,a,b)$
$\sigma_{A=B}(R_1 \times R_2)(a,a,c)$
$\sigma_{A=B}(R_1 \times R_2)(a,b,c)$
$\sigma_{A=B}(R_1 \times R_2)(b,b,c)$
$R_1 \times R_2(a,a,b)$
$R_1 \times R_2(a,a,c)$
$R_1 \times R_2(a,b,c)$
$R_1 \times R_2(b,a,b)$
$R_1 \times R_2(b,a,c)$
$R_1 \times R_2(b,b,c)$
$R_1(a)$ $R_1(b)$ $R_1(c)$ $R_2(a,a)$ $R_2(a,b)$ $R_2(a,c)$ $R_2(b,c)$
T T F F T T T


$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(a)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(b)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(c)$
and
not
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(a)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(b)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(c)$
or
...
$\sigma_{A=B}(R_1 \times R_2)(a,a,b)$
$\sigma_{A=B}(R_1 \times R_2)(a,a,c)$
$\sigma_{A=B}(R_1 \times R_2)(a,b,c)$
$\sigma_{A=B}(R_1 \times R_2)(b,b,c)$
$R_1 \times R_2(a,a,b)$
$R_1 \times R_2(a,a,c)$
$R_1 \times R_2(a,b,c)$
$R_1 \times R_2(b,a,b)$
$R_1 \times R_2(b,a,c)$
$R_1 \times R_2(b,b,c)$
$R_1(a)$ $R_1(b)$ $R_1(c)$ $R_2(a,a)$ $R_2(a,b)$ $R_2(a,c)$ $R_2(b,c)$
T T F F T T T
70


$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(a)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(b)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(c)$
and
and
and
not
not
not
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(a)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(b)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(c)$
or
or
or
...
...
...
$\sigma_{A=B}(R_1 \times R_2)(a,a,b)$
$\sigma_{A=B}(R_1 \times R_2)(a,a,c)$
$\sigma_{A=B}(R_1 \times R_2)(a,b,c)$
$\sigma_{A=B}(R_1 \times R_2)(b,b,c)$
...
$R_1 \times R_2(a,a,b)$
$R_1 \times R_2(a,a,c)$
$R_1 \times R_2(a,b,c)$
$R_1 \times R_2(b,a,b)$
$R_1 \times R_2(b,a,c)$
$R_1 \times R_2(b,b,c)$
and
and
and
and
and
and
$R_1(a)$
$R_1(b)$
$R_1(c)$
$R_2(a,a)$
$R_2(a,b)$
$R_2(a,c)$
$R_2(b,c)$
T
T
F
F
T
T
T

71


F
F
T
$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(a)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(b)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2)) - R_1(c)$
and
not
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(a)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(b)$
$\pi_C(\sigma_{A=B}(R_1 \times R_2))(c)$
or
...
$\sigma_{A=B}(R_1 \times R_2)(a,a,b)$
$\sigma_{A=B}(R_1 \times R_2)(a,a,c)$
$\sigma_{A=B}(R_1 \times R_2)(a,b,c)$
$\sigma_{A=B}(R_1 \times R_2)(b,b,c)$
$R_1 \times R_2(a,a,b)$
$R_1 \times R_2(a,a,c)$
$R_1 \times R_2(a,b,c)$
$R_1 \times R_2(b,a,b)$
$R_1 \times R_2(b,a,c)$
$R_1 \times R_2(b,b,c)$
$R_1(a)$
$R_1(b)$
$R_1(c)$
$R_2(a,a)$
$R_2(a,b)$
$R_2(a,c)$
$R_2(b,c)$
72
T
T
F
F
T
T
T

Remarks

- Schema and query are assumed fixed.
- Database and size of the active domain are variable.
- Uniform families of gates:
  Total number of input gates uniquely determines size of active domain.
- Example:Schema: $R(ABC), S(D), T(EF)$
  Number of input gates: $n^3 + n + n^2$ for some n (which is the size of the active domain).