Query languages (NDBI049)

Expressive power

Jaroslav Pokorný
MFF UK, Praha
jaroslav.pokorny@matfyz.cuni.cz

Content

1. Three semantics of domain relational calculus (DRC). Definite and safe formulas of DRC. Proof of the equivalence of the relational algebra (RA) and DRC restricted to definite formulas.
2. Transitive closure of a binary relation. Impossibility to express it in relational algebra.
3. Composition of RA expressions, the least fixpoint approach, minimal fixpoint.

DRC semantics (1)

Assumptions: query expressions $\{x_1,\ldots,x_k \mid A(x_1,\ldots,x_k)\}$, *A* is a DRC formula,

database $\mathbf{R}^*$, *dom* is domain for **R**; *actual domain* of formula *A*, *adom*(*A*), is a set of values from relations in *A* and constants in *A*.

Three problems:

- potential possibility of infinite answer (in the case of infinite *dom*)
- situation, when TRUE-assignment of free variables is not from $\mathbf{R}^*$.
- how to implement evaluation of a quantification (in the case of infinite *dom*) in a finite time.

DRC semantics (2)

3 semantics of DRC, solving the problems:

(i) unlimited interpretation with restricted output

(ii) limited interpretation

(iii) domain-independent queries

Notation: result of a query *Q* evaluation in the unlimited interpretation as $Q^{dom}[\mathbf{R}^*]$.

Then:

- for (i) the result is defined as $Q^{dom}[\mathbf{R}^*] \cap adom^k$, where *k* is order of the resulted relation.
- for (ii) variables ranges over *adom*, i.e. $Q^{adom}[\mathbf{R}^*]$.

DRC semantics (3)

Q.: $\{x \mid \neg R(A{:}x)\}$

$\Rightarrow$ The answer depends on *dom*(*A*).

$\Rightarrow$ Query expression defines different queries for different domains.

Remark: A query, returning $\emptyset$, can be domain dependent in the case the quantified variable ranges over an infinite set, e.g.

Q.: $\{x \mid \forall\, y\, \mathrm{R}(x,y)\}$

Df.: We say that a query expression is *domain-independent* (*definite*) if the answer to it does not depend on *dom*.

Query language is *domain-independent*, if each its expression is domain-independent. The result of *Q* is equal to $Q^{\mathrm{dom}}[\mathbf{R}^*] = Q^{\mathrm{adom}}[\mathbf{R}^*]$.

BOOK(ISBN,TITLE, AUTHOR)
COPY(C_Number, ISBN)
LOAN(C_Number, Cust_Number, DueDate)

DRC semantics (4)

⇒ evaluation of a domain independent expression in unlimited interpretation returns the same result as in restricted interpretation.

Ex.: $\neg$ BOOK(TITLE:'Introduction to DBS', AUTHOR:a)

IS NOT definite.

$\exists$cn (COPY(cn,i) $\wedge$ LOAN(cn,b,dd))

IS definite

$\exists$cn (COPY(cn,d) $\vee$ LOAN(cn,b,dd))

IS NOT definite, if variables are untyped or of too "wide" types

Theorem (Di Paola 1969): Definiteness of A is not decidable.

⇒ The language of domain-independent expressions is not decidable.

Remark: Relational algebra is a domain-independent language.

DRC semantics (5)

Notation of DRC:

- in unlimited interpretation with restricted output DRC^{rout},
- in limited interpretation DRC^{lim}
- domain independent expressions DRC^{ind}.

Statement: $DRC^{rout} \cong DRC^{lim} \cong DRC^{ind}$. Moreover,

(i) if *Q* is a DRC expression, then there is a domain independent expression Q‘, which after evaluation returns the same result as Q in unlimited interpretation with restricted output.

(ii) if *Q* is a DRC expression, then there is a domain independent expression Q‘, which after evaluation returns the same result as Q in limited interpretation.

DRC semantics (6)

Proof (sketch): trivially DRC^{rout} and DRC^{lim} are at least so powerful as DRC^{ind}, i.e. DRC^{ind} < DRC^{lim} and DRC^{rout} < DRC^{lim}

- We show a power of DRC^{lim}

If $Q \in$ DRC^{ind}, then it returns $Q^{dom}[\mathbf{R}^*]$, přičemž $Q^{dom}[\mathbf{R}^*] = Q^{adom}[\mathbf{R}^*]$.

Let $Q \in$ DRC. Then it is possible to construct Q‘ so that all free and bound variables in the formula of query *Q*‘ are restricted to the active domain. Then $D'^{adom}[\mathbf{R}^*] = D^{adom}[\mathbf{R}^*]$. Expression *Q*‘ is however domain independent, so DRC^{lim} < DRC^{ind}. We also demonstrated the (ii) part of the statement. Thus DRC^{lim} $\cong$ DRC^{ind}.

- It holds, that DRC^{rout} is more powerful than DRC^{lim}. A proof of (i) is technically more complicated (see [Hull and Su 94]).

Safe formulas in DRC

Df.: A safe DRC formula, *A*, is a DRC formula, which is definite and syntactically characterizable.

1. $\forall$, $\Rightarrow$ are eliminated
2. if *A* contains a disjunction, then is it is a subformula

   $\varphi_1(x_1,\ldots,x_s) \vee \varphi_2(x_1,\ldots,x_s)$,

   i.e. φ_i contain the same free variables,
3. if A contains a conjunction (maximal), e.g.,

   $\varphi \equiv \varphi_1 \wedge \ldots \wedge \varphi_r$, $r \geq 1$, then each free variable in φ is limited, i.e., at least one of the following conditions holds:

   - A variable is free in φ_i , which is neither arithmetic comparison and nor negation,
   - there is $\varphi_i \equiv x=a$, where a is a constant,
   - there is $\varphi_i \equiv x=y$, where y is limited.

Safe formulas in DRC

4. $\neg$ can be used only in conjunctions of type 3.

Remarks:

- Any safe formula is definite.
- There are definite formulas which are not safe.

Ex.:

$x=y$ IS NOT safe

$x=y \vee R(x,y)$ IS NOT safe

$x=y \wedge R(x,y)$ IS safe

$R(x,y,z) \wedge \neg (P(x,y) \vee Q(y,z))$ IS NOT safe, is definite.

$R(x,y,z) \wedge \neg P(x,y) \wedge \neg Q(y,z)$ IS safe!

Equivalence of relational languages

4 approaches:

- domain relational calculus (DRC)
- tuple relational calculus (NRC)
- relational algebra (A_R)
- DATALOG

We prove: DRC $\cong A_R$

Lemma: Let φ be a Boolean expression created by using $\neg$, $\wedge$, $\vee$ and simple selections X θ Y or X θ k, where $\theta \in \{\leq, \geq, \neq, <, >, =\}$, k is constant and X, Y are attribute names. Then for E(φ), where E $\in A_R$, there is a relational expression E', whose each selection is simple and E(φ) $\cong$ E'.

Proof: 1. each $\neg$ is propagated to a simple selection and θ is replaced by its negation.

Equivalence of relational languages

2. by induction on the number of operators $\wedge$, $\vee$.

for $\varnothing$ of operators - trivial

$E(\varphi) \equiv E(\varphi_1 \wedge \varphi_2)$ and E contains at most selections, which are simple. Then $E(\varphi) \equiv E(\varphi_1)(\varphi_2)$.

$E(\varphi) \equiv E(\varphi_1 \vee \varphi_2)$ and E contains at most selections, which are simple. Then $E(\varphi) \equiv E(\varphi_1) \cup E(\varphi_2)$.

Ex.: $E \equiv R(\neg (A_1 = A_2 \wedge (A_1 < A_3 \vee A_2 \leq A_3)))$

then $\varphi \equiv A_1 \neq A_2 \vee (A_1 \geq A_3 \wedge A_2 > A_3)$

and $E' \equiv R(A_1 \neq A_2) \cup R(A_1 \geq A_3)(A_2 > A_3)$

From relational algebra to DRC

Theorem: Each query expressible in A_R is expressible in DRC.

Proof: by induction on the number of operators in relational expression E.

1. $\emptyset$ operators in E.

   $E \equiv R \rightarrow \{x_1,\ldots,x_k \mid R(x_1,\ldots,x_k)\}$

   $E \equiv$ const. relation $\rightarrow \{x_1,\ldots,x_k \mid x_1 = a_1 \wedge \ldots \wedge x_k = a_k \vee$

   $x_1 = b_1 \wedge \ldots \wedge x_k = b_k \vee \ldots\}$

2. $E \equiv E_1 \cup E_2$ by the induction hypothesis there are formulas e_1 and e_2 with free variables $x_1,\ldots,x_k$

   $\rightarrow \{x_1,\ldots,x_k \mid e_1(x_1,\ldots,x_k) \vee e_2(x_1,\ldots,x_k)\}$

From relational algebra to DRC

3. $E \equiv E_1 - E_2$

$\rightarrow \{x_1,\ldots,x_k \mid e_1(x_1,\ldots,x_k) \wedge \neg e_2(x_1,\ldots,x_k)\}$

4. $E \equiv E_1[i_1,\ldots,i_k]$

$\rightarrow \{x_{i1},\ldots,x_{ik} \mid \exists x_{j1},\ldots,x_{j(n-k)}\, e_1(x_1,\ldots,x_n)\}$

5. $E \equiv E_1 \times E_2$

$\rightarrow \{x_1,\ldots,x_m\, x_{m+1},\ldots,x_{m+n} \mid e_1(x_1,\ldots,x_m) \wedge e_2(x_{m+1},\ldots,x_{m+n})\}$

6. $E \equiv E_1\,(\varphi)$

$\rightarrow \{x_1,\ldots,x_k \mid e_1(x_1,\ldots,x_k) \wedge x_A\, \theta\, x_B)\}$, if $\varphi \equiv A\, \theta\, B$

or $x_A\, \theta\, a$ if $\varphi \equiv A\, \theta\, a$

By lemma, it is enough, when φ denotes a simple selection.

Semantic definition of definite formulas

Sufficient conditions for definite formulas *A*:

1. components of TRUE-assignment of *A* are from *adom*(*A*).
2. if $A' \equiv \exists y\ \varphi(y)$, then if for a y_0

   $\varphi(y_0) \Leftrightarrow$ TRUE, then $y_0 \in$ adom(φ).
3. if $A' \equiv \forall y\ \varphi(y)$, then if for a y_0

   $\varphi(y_0) \Leftrightarrow$ FALSE, then $y_0 \in$ adom(φ).

Remark: 2. and 3. holds for any allowable values of free variables in φ (except y).

Remark: explanation of condition 3.

$\forall y\ \varphi(y) \Leftrightarrow \neg\exists y\ \neg\varphi(y)$

$\Rightarrow$ if for a y_0 $\neg\varphi(y_0) \Leftrightarrow$ TRUE, then by 2., $y_0 \in$ adom($\neg\varphi$).

Semantic definition of definite formulas

Since adom($\neg\varphi$) = adom(φ), then

$\varphi(y_0) \Leftrightarrow$ FALSE $\Rightarrow y_0 \in$ adom(φ).

Statement: Elimination of $\forall$ and $\wedge$ from a definite formula leads to a definite formula as well.

From DRC to relational algebra

Statement: Each query expressible by a definite expression of DRC is expressible in A_R.

Proof: by induction on the number of operators in A of the definite expression $\{x_1,\ldots,x_k \mid A(x_1,\ldots,x_k)\}$ (+)

- We express *adom*(*A*) as expression A_R. We denote it as E .
- We alter *A*, that it contains only $\exists$, $\vee$, $\neg$.
- The proof will be done for *adom*$(A)^k \cap \{x_1,\ldots,x_k \mid A'(x_1,\ldots,x_k)\}$. When $A' \equiv A$ and A is definite, $\cap$ leads to expression (+).

By induction:

1. $\varnothing$ of operators in A'. Then A' is atomic formula.

   $x_1\ \theta\ x_2 \rightarrow (E \times E)(1\ \theta\ 2)$

   $x_1\ \theta\ a \rightarrow E(1\ \theta\ a)$

   $R(x_1,\ldots,x_m) \rightarrow R(\ldots \wedge i_1\ \theta\ i_2 \wedge \ldots)[\ldots, i_1,\ldots]$, when, e.g., $x_{i1} = x_{i2}$

From DRC to relational algebra

2. A' has at least one operator and the induction hypothesis holds for all subformulas from A' with less operators than A'.

- $A'(u_1,...,u_m) \equiv A^1(u_1,...,u_n) \vee A^2(u_1,...,u_p)$. Then for expressions $\mathit{adom}(A)^m \cap \{\underline{u}|A^i(\underline{u})\}$ there are relational expressions E_i. Transformation leads to $\cup$.

  Ex.: $A'(u_1,u_2,u_3,u_4) \equiv A^1(u_1,u_3,u_4) \vee A^2(u_2,u_4)$

  $$\rightarrow (E_1 \times E)\ [1,4,2,3] \cup (E_2 \times E \times E)\ [3,1,4,2]$$

- $A'(u_1,...,u_m) \equiv \neg A^1(u_1,...,u_m)$. Then for expression $\mathrm{adom}(A)^m \cap \{\underline{u}|A^1(\underline{u})\}$ there is a relational expression E_1. Transformation leads to -, i.e., $E^m - E_1$

- $A'(u_1,...,u_m) \equiv \exists u_{m+1} A^1(u_1,...,u_m, u_{m+1})$. Then for expression $\mathrm{adom}(A)^{m+1} \cap \{\underline{u}|A^1(\underline{u})\}$ there is a relational expression E_1. Transformation leads to [], i.e. $E_1[1,2,...,m]$.

If $A' \equiv A$, then the answer is not changed.

From DRC to relational algebra

Ex.: $\{w,x \mid R(w,x) \wedge \forall y(\neg S(w,y) \wedge \neg S(x,y))\}$ is a definite expression.

Justification: $dom(\neg S(w,y) \wedge \neg S(x,y)) = dom(S)$

Let $y_0 \notin dom(S)$. Then $\neg S(w,y_0) \wedge \neg S(x,y_0) \Leftrightarrow TRUE$.

So, the condition 3 from sufficient conditions is fulfilled

Eliminating $\wedge$ and $\forall$, we obtain the definite expression:

$\{w,x \mid \neg (\neg R(w,x) \vee \exists y(S(w,y) \vee S(x,y))\}$

Transformation:

$S(w,y) \vee S(x,y) \rightarrow (S \times E)[1,3,2] \cup (S \times E)[3,1,2]$

$\exists y$ (-”-) $\rightarrow$ (-”-) [1, 2] we denote as E’

Remark: E’ can be optimized as $(S \times E)[1,3] \cup (S \times E)[3,1]$

$\neg R(w,x) \rightarrow E^2 - R$

$\neg R(w,x) \vee \exists y(S(w,y) \vee S(x,y) \rightarrow (E^2 - R) \cup E'$

$\neg$ (-”-) $\rightarrow E^2 - ((E^2 - R) \cup E')$

From DRC to relational algebra

Problem: the result leads to a non-effective evaluation

Optimization:

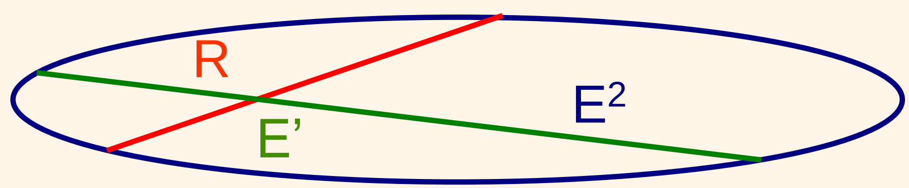
Let $\underline{X}$ denote the complement of X w.r.t. E.

It holds: $\underline{X \cup Y} = \underline{X} \cap \underline{Y}$

$\Rightarrow E^2 - ((E^2 - R) \cup E') = (E^2 - (E^2 - R)) \cap (E^2 - E')$

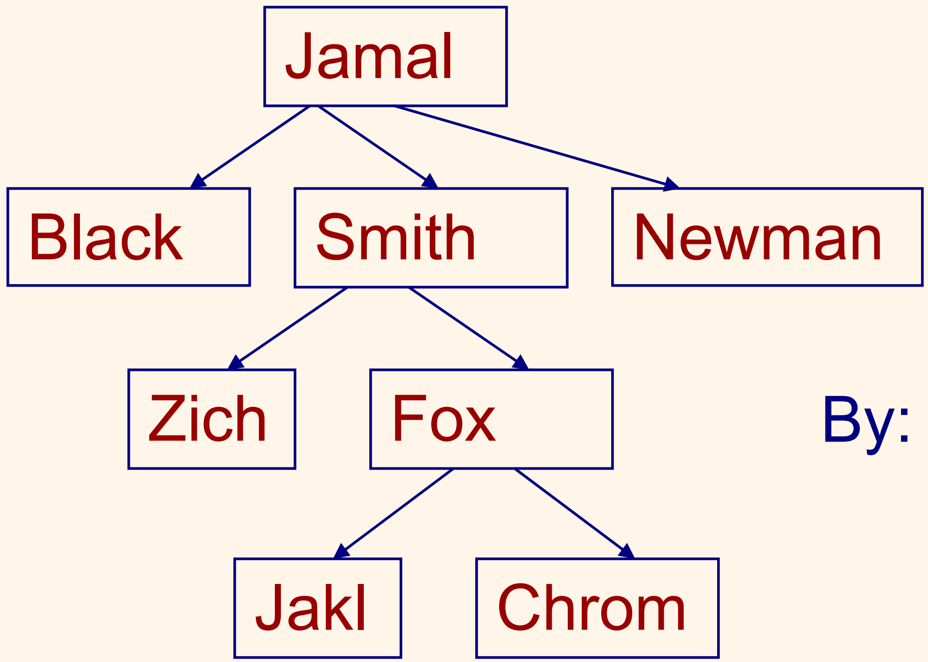
$= R \cap \underline{E'} = R - E'$

Visualization:



Power of DRC (A_R)

Q.: Find all subordinates of Smith.



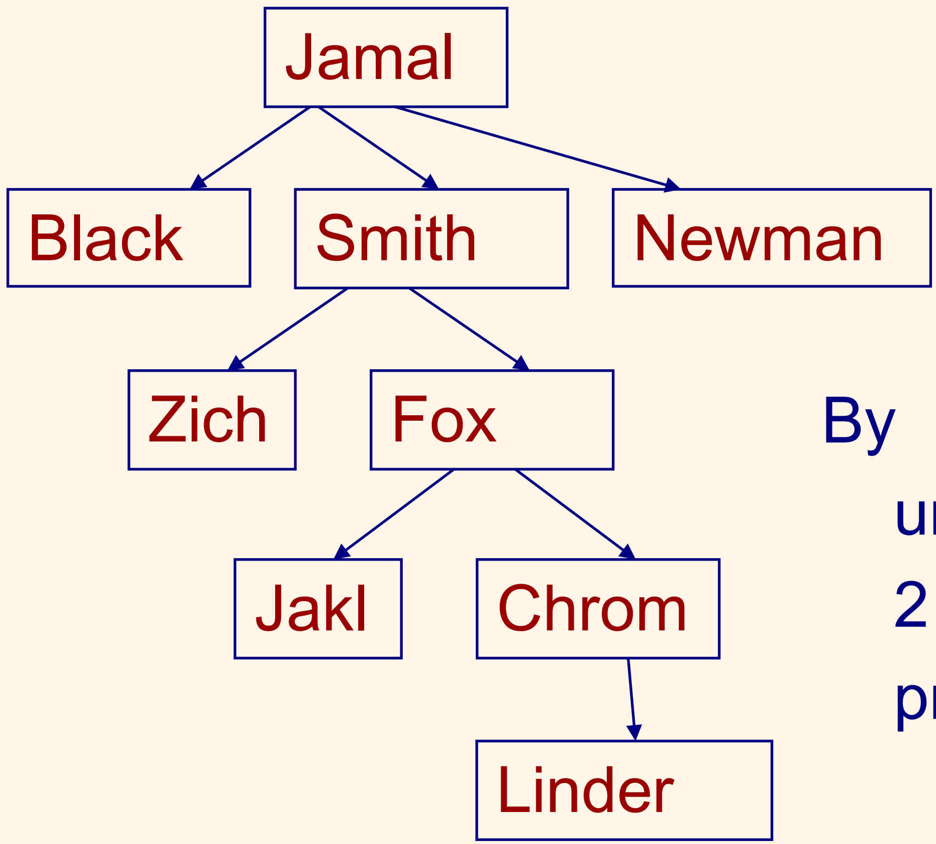
By: union

1 join

projection

Power of DRC (A_R)

Q.: Find all subordinates of Smith.



By

union

2 joins

projection

Query transitive closure (0)

Notions:

Df.: Binary relation R is transitive, if for each $(a,b)\in R$ and $(b,c)\in R$ also $(a,c)\in R$.

Df.: Transitive closure of the relation R, R^+, is least transitive relation containing R.

Database notions: relation schema R, relation R^*

Ex.: SUP-SUB(Superior, Subordinate) reflects transitive relationships on a conceptual level.

SUP-SUB* contains only direct relationships, e.g. (Jamal, Smith), (Fox, Chrom), …

Goal: calculate transitive closure of the relation SUP-SUB*

Assumption: We will consider relations, which are transitive on a conceptual level.

Query transitive closure (1)

Statement: Let R be a binary relation schema. Then there is no expression A_R, calculating for each relation R* its transitive closure R^+.

Proof:

1. Consider $\Sigma_s = \{a_1, a_2, \ldots, a_s\}$, $s \geq 1$, as a set of constants, for which no ordering exists, and

$$R_s = \{a_1 a_2,\ a_2 a_3, \ldots, a_{s-1} a_s\}$$

Remark: $R_s \Leftrightarrow$ graph $a_1 \rightarrow a_2 \rightarrow \ldots \rightarrow a_s$, i.e., transitivity is defined by connectivity in a directed graph.

Remark: if an ordering $<$ is defined on Σ_s, then

$R_s^+ \cong (R_s[1] \times R_s[2])(1 < 2)$

2. We show, that for arbitrary expression E(R) there is *s* such, that $E(R_s) \neq R_s^+$.

Query transitive closure (2)

3. Lemma: Let E be a relational algebra expression. Then for sufficiently big *s*

   $E(R_s) \cong \{b_1,\ldots,b_k \mid \Gamma(b_1,\ldots,b_k)\}$,

where $k \geq 1$ and Γ is a formula in a disjunctive normal form.

Atomic formulas in Γ have a special form:

   $b_i = a_j$, $b_i \neq a_j$,

   $b_i = b_j + c$ or $b_i \neq b_j + c$, where c is (not necessarily positive) constant, where $b_j + c$ is abbreviation for "such a_m, for which $b_j = a_{m-c}$"

Domain of interpretation for assignments to variables b_j is Σ_s.

Remark: $b_i = b_j + c \Leftrightarrow b_i$ is behind b_j in the distance *c* nodes.

Query transitive closure (3)

4. Proof by contradiction.

There is E such, that $E(R) = R^+$ and any relation R, i.e. also $E(R_s) = R_s^+$ for sufficiently big *s*

- by lemma, $R_s^+ \cong \{b_1,b_2 \,|\Gamma(b_1,b_2)\}$

There are two cases:

(a) each clause z Γ contains an atom of form

$b_1 = a_i$, $b_2 = a_i$ or $b_1 = b_2 + c$ ($\Leftrightarrow b_2 = b_1 - c$)

Let $b_1 b_2 = a_m a_{m+d}$,

where $m >$ arbitrary i and $d >$ arbitrary c

Query transitive closure (4)

$\Rightarrow$ $b_1 = a_m$ and $b_2 = a_{m+d}$ do not meet any clause from Γ.

$\Rightarrow$ contradiction ($a_m a_{m+d} \notin R_s^+$)

(b) in Γ there are clauses with atoms containing only $\neq$.

Let $b_1 b_2 = a_{m+d} a_m$, where neither $b_i \neq a_m$ nor $b_i \neq a_{m+d}$ is contained in Γ, and $d > 0$ is greater than arbitrary c in $b_1 \neq b_2 + c$ or $b_2 \neq b_1 + c$ in Γ (see construction of Γ)

$\Rightarrow$ $a_{m+d}\, a_m \in E(R_s)$ for sufficient s, but $\notin R_s^+$ $\Rightarrow$ contradiction

Thus: for arbitrary expression E, always there is *s* for which

$$E(R_s) \neq R_s^+$$

Query transitive closure (5)

5. Proof of lemma – by induction on the number of operators in E

I. $\varnothing$ of operators $\Rightarrow E \equiv R_s$ or E is a constant relation

$\Rightarrow E \equiv \{b_1,b_2 \mid b_2 = b_1 + 1\}$ and

$E \equiv \{b_1 \mid b_1 = c_1 \vee b_1 = c_2 \vee \ldots \vee b_1 = c_m\}$, respectively

II. a) $E \equiv E_1 \cup E_2$, $E_1 - E_2$, $E_1 \times E_2$

$E_1 \cong \{b_1,\ldots,b_k \mid \Gamma_1(b_1,\ldots,b_k)\}$

$E_2 \cong \{b_1,\ldots,b_m \mid \Gamma_2(b_1,\ldots,b_m)\}$

$\Rightarrow$ for $\cup$ and - k=m and therefore

Query transitive closure (6)

$E \cong \{b_1,\ldots,b_k \mid \Gamma_1(b_1,\ldots,b_k) \vee \Gamma_2(b_1,\ldots,b_k)\}$,

$E \cong \{b_1,\ldots,b_k \mid \Gamma_1(b_1,\ldots,b_k) \wedge \neg\, \Gamma_2(b_1,\ldots,b_k)\}$, respectively.

$\Rightarrow$ for $\times$

$E \cong \{b_1,\ldots,b_k\ b_{k+1},\ldots,b_{k+m} \mid \Gamma_1(b_1,\ldots,b_k) \wedge \Gamma_2(\ b_{k+1},\ldots,b_{k+m})\}$

!! Then a transformation to DNF follows.

b) $E \equiv E_1(\varphi)$ a φ contains either = or $\neq$

$\Rightarrow E \cong \{b_1,\ldots,b_k \mid \Gamma_1\ (b_1,\ldots,b_k) \wedge \varphi(b_1,\ldots,b_k)\}$

Query transitive closure (7)

c) $E \equiv E_1[S]$

We will consider a projection removing one attribute

$\Rightarrow$ It is about a sequence of permutations of variables and elimination of the last component.

The elimination of b_k leads to

$$\{b_1,\ldots,b_{k-1} | \exists\, b_k\, \Gamma(b_1,\ldots,b_k)\}$$, where Γ is in DNF

$\Rightarrow$ by a)

$$\cup_{i=1..m}\{b_1,\ldots,b_{k-1} \mid \exists b_k\, \Gamma_i(b_1,\ldots,b_k)\}$$

$\Rightarrow$ we will eliminate $\exists$ from one conjunct

- in Γ_i there are not $b_k=a_i$, $b_i=b_k+c$, and $b_k=b_i+c$

$\Rightarrow$ $\{b_1,\ldots,b_{k-1} \mid \underline{\Gamma}_i(b_1,\ldots,b_{k-1})\}$

where $\underline{\Gamma}_i$ does not contain $b_k \neq a_i$, $b_i \neq b_k+c$, or $b_k \neq b_i+c$

Query transitive closure (8)

- in Γ_i there is either $b_k=a_i$ or $b_i=b_k+c$ or $b_k=b_i+c$
  $\Rightarrow$ substitutions for b_k will take place.
  The results are adjusted to TRUE
  or FALSE
  or $b_t=b_j+g$
  and the following inequalities are added:
  $b_i \neq a_j$ for $s\text{-}c < j \leq s$,
  $b_i \neq a_j$ for $1 < j \leq c$, respectively

Transitive closure functionally

Df.: A composition R ° S of binary relations R, S defined on domain D is a binary relation

$$\{a,b \mid \exists c \in D, (a,c) \in R^* \wedge (c,b) \in S^* \}$$

Let f be a function assigning to a binary relation R a binary relation R´ (both relations are defined on D).

Df.: Let R be relational variable and f(R) relational expression. Then the least fixpoint (LFP) of the equation

$$R = f(R) \qquad (1)$$

is a relation R* such, that:

- $R^* = f(R^*)$ */fixpoint/*
- $S^* = f(S^*) \Rightarrow R^* \subseteq S^*$ */minimality/*

Df.: f is monotonic if for each two relations R^*_1 and R^*_2

$$R^*_1 \subseteq R^*_2 \Rightarrow f(R^*_1) \subseteq f(R^*_2)$$

Transitive closure functionally

Statement: f is monotonic if and only if

$$f(R_1 \cup R_2) \supseteq f(R_1) \cup f(R_2)$$

Df.: f is additive if and only if

$$f(R_1 \cup R_2) = f(R_1) \cup f(R_2)$$

Statement: Additive function is monotonic.

Theorem (Tarski): If f is monotonic, then the LFP of equation (1) exists.

LFP construction: For a finite relation R, we obtain LFP by repeating application of f.

Initialize R by $\emptyset$, then $f^{i-1}(\emptyset) \subseteq f^{i}(\emptyset)$

Then there is $n_0 \geq 1$ such that

$$\emptyset \subset f(\emptyset) \subset f^1(\emptyset) \subset \dots \subset f^{n0}(\emptyset) = f^{n0+1}(\emptyset)$$

Relation $f^{n0}(\emptyset)$ is the LFP of the equation (1).

Transitive closure functionally

Proof: By induction on *i*, it is shown, that relation $f^{n0}(\emptyset)$ is contained in each fixpoint of equation (1).

Statement: The transitive closure of a binary relation R* defined on D is the LFP of the equation

$$S = S \circ R^* \cup R^*$$

where S is a relational variable (binary, defined on D).

Proof: $f(S) = S \circ R^* \cup R^*$

$\Rightarrow \quad f^n(\emptyset) = \cup_{i=1..n} R^* \circ R^* \circ \ldots \circ R^*$

which leads to the transitive closure

$$\cup_{i = 1..\infty} R^* \circ R^* \circ \ldots \circ R^*$$

Transitive closure functionally

Ex.: Consider the relation schema

FLIGHTS(FROM, TO, DEPARTURE, ARRIVAL)

Task: to express CONNECTIONS with transfers

Solution: CONNECTIONS* is given as the LFP of equation

$$\text{CONNECTIONS} = \text{FLIGHTS} \cup (\text{FLIGHTS} \times \text{CONNECTIONS})(2{=}5 \wedge 4 < 7)[1, 6, 3, 8]$$

Statement: Each relational algebra expression not containing difference is additive in all its variables.

Transitive closure functionally

Remarks:

- Non-monotonic expression can have an LFP,
- Not every expression involving the difference operator fails to be monotone.

Df.: A minimální fixpoint (MFP) of equation (1) is such fixpoint R*, that there is no other fixpoint, which is a proper subset of R*.

$\Rightarrow \exists$ LFP, then it is the only one MFP.

If there is more MFPs, then they are mutually non-comparable and no LFP exists.

Databases intensionally

Ex.: Consider predicates

F(x,y) x is a father of y

M(x) x is a man

S(x,y) x is a sibling of y

B(x,y) x is a brother of y

Extensional database (EDB):

F(James, Paul) (1)

F(James, Jerry) (2)

F(Jerry, Veronika) (3)

Databases intensionally

Intensional database (IDB):

M(x):- F(x,y) (4)

S(y,w) :- F(x,y), F(x,w) (5)

B(x,y) :- S(x,y),M(x) (6)

Queries:

Q_1: Has Paul a brother?

Q_2: Find all (x,y), where x is a brother of y.

Q_3: Find all (x,y), where x is a sibling of y.

Remark: EDB + IDB create a logical program (LP)

Solution of LP by the resolution method

EDB as a set of facts

IDB as a set Horn clauses:

$F(x,y) \Rightarrow M(x)$

$F(x,y) \wedge F(x,w) \Rightarrow S(y,w)$

$S(x,y) \wedge M(x) \Rightarrow B(x,y)$

Assumption: Formulas in IDB are universally quantified, e.g.,

$\forall x \forall y \forall w \ (F(x,y) \wedge F(x,w) \Rightarrow S(y,w))$

Reformulation of Q_1 : $\exists z \ B(z,Paul)$

Solution of LP by the resolution method

Resolution method:

- Uses a proof by contradiction
- inference is equivalent to deriving an empty clause (NIL); in other cases it is not possible to say, whether the clause is derivable

Principle: $A_1 \vee \ldots \vee A_i \vee B_1$ $\quad\quad\quad\quad$ $C_1 \vee \ldots \vee C_j \vee \neg B_2$

- Unification: by substitutions we try to achieve to do B_1 and B_2 complementary.
- Deriving a resolvent: If after unification the input has a form $\underline{A}_1 \vee \ldots \vee \underline{A}_i \vee B$ and $\underline{C}_1 \vee \ldots \vee \underline{C}_j \vee \neg B$, then it is possible to derive $\underline{A}_1 \vee \ldots \vee \underline{A}_i \vee \underline{C}_1 \vee \ldots \vee \underline{C}_j$

Solution of LP by the resolution method

Statement: A resolvent is (un)satisfiable, if input clauses were (un)satisfiable.

The procedure goal: to derive NIL

A is a logical conseqence of W

Justification: $W=\{A_1,\ldots,A_m\}$, then $W \models A$ if and only if

$A_1 \wedge \ldots \wedge A_m \wedge \neg A$ is unsatisfiable

By the Gödel theorem, unsatisfiability is partially decidable, i.e. there is a procedure P such that for each formula φ the following holds:

if φ is unsatisfiable, then $P(\varphi)$ terminates and announces it,

if φ is satisfiable, then $P(\varphi)$ either terminates and announces it, or fails to terminate.

Solution of LP by the resolution method

Ex.: We add to EDB and IDB $\neg$B(z,Paul) (7)

and run the resolution method:

(8) S(Jerry,w) :- F(James,w)	from (2),(3)
(9) S(Jerry,Paul)	from (8),(1)
(10) M(Jerry)	from (3),(4)
(11) B(Jerry,y) :- S(Jerry,y)	from (10),(6)
(12) B(Jerry,Paul)	from (11),(9)
(13) NIL	from (12),(7)