Rachunki relacji

Tadeusz Pankowski
www.put.poznan.pl/~tadeusz.pankowski

Rachunki relacji

1. RRK – Relacyjny Rachunek Krotek
2. RRD – Relacyjny Rachunek Dziedzin
3. Datalog – Database Prolog

RRK – Relacyjny Rachunek Krotek

1. Stanowi teoretyczną podstawę języka SQL.
2. Jest językiem rachunku predykatów pierwszego rzędu i można go wykorzystywać do formułowania zapytań do relacyjnej bazy danych.
3. Definicja wymaga podania
   - składni: alfabetu i reguł tworzenia wyrażeń
   - semantyki: przypisanie wyrażeniom wartości w pewnej dziedzinie interpretacji.
4. Każda wartość przypisana wyrażeniu zwanemu *zapytaniem* traktowana jest jako odpowiedź na to zapytanie.

RRK jako język zapytań – przykład

DOSTAWCA

	NrDcy	Nazwa	Adres
d	10	X	A
	11	Y	A
	12	Z	B
	13	W	C

DOSTAWA

	NrDcy	NrTow	NrMag	Data	Ilość
w	10	100	200	a	15
	10	150	200	a	20
	12	100	201	b	30
	11	150	202	c	10

1. "Podaj nazwy dostawców o adresie B, którzy dostarczali towary o numerze 100 w ilości większej niż 10"

$$\{[\text{Nazwa} : d.\text{Nazwa}] \mid \text{DOSTAWCA}(d) \wedge d.\text{Adres} = \text{'B'} \wedge$$
$$\wedge \exists w\,(\text{DOSTAWA}(w) \wedge w.\text{NrTow} = 100 \wedge w.\text{Ilość} > 10 \wedge$$
$$\wedge d.\text{NrDcy} = w.\text{NrDcy})\}.$$

RRK - alfabet

1. Symbole specyficzne (pozalogiczne)
   a) stałe proste: c_1, c_2, ...
   b) symbole funkcyjne: $.A_1$, $.A_2$, ...
      (dla każdego atrybutu A istnieje symbol funkcyjny .A)
   c) nazwy tabel (symbole relacyjne): R_1, R_2, ...
2. Symbole ogólne (wspólne dla wszystkich języków)
   a) zmienne krotkowe: v_1, v_2, ...
   b) symbole porównań: $=, \neq, <, \leq, >, \geq$
   c) symbole logiczne: $\neg, \vee, \wedge, \Rightarrow, \exists, \forall$
   d) ograniczniki i nawiasy: , | { } ( )

Z nazwą tabeli *R* związany jest jej *typ* rozumiany jako skończony zbiór atrybutów.

RRK – reguły tworzenia wyrażeń

1. **Termy**:
   $a ::= c \mid v.A$
2. **Formuły**:
   $\varphi ::= R(v) \mid a_1\ \theta\ a_2 \mid (\varphi) \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid \exists v\, \varphi \mid \forall v\, \varphi$,
   (v musi być zmienną wolną w φ).
3. **Zapytania**:
   $\{[A_1 : a_1, ..., A_n : a_n] \mid \varphi\}$,
   (zbiór zmiennych występujących w *krotce wynikowej* $[A_1 : a_1, ..., A_n : a_n]$ musi być równy zbiorowi zmiennych wolnych w *kwalifikatorze* φ).

Semantyka – dziedzina interpretacji

Dziedzina interpretacj dla RRK (zbiór atrubutów nieskończony):

$$DZ_{RRK} = (Val \cup Tup \cup Tab, \{f_{A_1}, f_{A_2}, \ldots\})$$

- Val – zbiór wartości proste,
- Tup – zbiór krotek,
- Tab – zbiór tabel,
- f_A : Tup → Val – funkcja, która zwraca wartość atrybutu A w podanej krotce,

  $f_{Kierunek}$([Nazwisko:"Kowalski", Kierunek:"Inf"]) = "Inf"

Semantyka RRK – interpretacja symboli pozalogicznych alfabetu i wartościowanie zmiennych

$$DZ_{RRK} = (Val \cup Tup \cup Tab, \{f_{A_1}, f_{A_2}, \ldots\})$$

Interpretacja symboli pozalogicznych – funkcja I:

- $c^I \in$ Val – stałym prostym odpowiadają wartości proste,
- $.A^I = f_A$ – symbolom funkcyjnym przypisywane są funkcje,
- $R^I \in$ Tab – nazwom tabel przypisywane są tabele (tego samego typu co nazwa tabeli).

Wartościowanie zmiennych – funkcja ω:

$$\omega : Var \rightarrow Tup$$

Semantyka RRK – obliczanie wyrażeń

Wartość termów (przy interpretacji $.^I$ i wartościowaniu ω:

$$(v.A)^I(\omega) = f_A(\omega(v))$$

Spełnianie formuł (przy interpretacji $.^I$ i wartościowaniu ω.

Przykład: Formuła DOSTAWA(w) $\wedge$ w.NrTow = 100 jest spełniona dla (I, ω), jeśli krotka przypisana zmiennej w należy do tabeli przypisanej nazwie DOSTAWA i pole NrTow w tej krotce ma wartość 100.

Formalnie:

(I, ω) |= (DOSTAWA(w) $\wedge$ w.NrTow = 100) w.i t.w. gdy

$\omega(w) \in$ DOSTAWA^I i $f_{NrTow}(\omega(w)) = 100$.

RRK – spełnianie formuł

Spełnianie formuł (obliczanie wartości logicznej formuł),
$(I, \omega) \models \varphi$ – spełnianie formuły φ przy interpretacji I i wartościowaniu ω:

$(I, \omega) \models R(v)$	$\Leftrightarrow$	$\omega(v) \in R^I$,
$(I, \omega) \models a\ \theta\ a'$	$\Leftrightarrow$	$\omega(a)\ \theta\ \omega(a')$,
$(I, \omega) \models (\varphi)$	$\Leftrightarrow$	$(I, \omega) \models \varphi$,
$(I, \omega) \models (\varphi \vee \varphi')$	$\Leftrightarrow$	$(I, \omega) \models \varphi \vee (I, \omega) \models \varphi'$,
$(I, \omega) \models (\varphi \wedge \varphi')$	$\Leftrightarrow$	$(I, \omega) \models \varphi \wedge (I, \omega) \models \varphi'$,
$(I, \omega) \models (\exists v\ \varphi)$	$\Leftrightarrow$	istnieje taka krotka t, że $(I, \omega[v \to t]) \models \varphi$,
$(I, \omega) \models (\forall v\ \varphi)$	$\Leftrightarrow$	dla każdej krotki t, $(I, \omega[v \to t]) \models \varphi$.

$\omega[v \to t]$ – wartościowanie, które zmiennej v przypisuje krotkę t.

RRK – odpowiedzi na zapytanie

1. $\{t \mid \varphi\}$ – zapytanie wyrażone w RRK.
2. Odpowiedzią na zapytanie $\{t \mid \varphi\}$ nazywamy każdą krotkę $\omega(t)$, określoną przez wartościowanie ω, dla którego spełniony jest kwalifikator φ zapytania, tj. $(I, \omega) \models \varphi$. Szukanie odpowiedzi jest więc równoważne szukaniu właściwych wartościowań zmiennych.
3. Zbiór wszystkich odpowiedzi na zapytanie $\{t \mid \varphi\}$:

$$\{t \mid \varphi\}^I = \{\omega(t) \mid (I, \omega) \models \varphi\}.$$

W praktyce:

R^I – tabela w bazie danych;

$\omega(v)$ – wybór krotki z tabeli i przypisanie jej zmiennej v.

RRK jako język zapytań - przykład

1. Wartości proste: 10, X, A, 11, 12, ...
2. Atrybuty: NrDcy, Nazwa, Adres, NrTow, NrMag, Data, Ilość.
3. Nazwy relacji i ich typy:

DOSTAWCA	typu {NrDcy,Nazwa,Adres}
TOWAR	typu {NrTow,GrupaTow}
MAGAZYN	typu {NrMag,Adres}
DOSTAWA	typu {NrDcy,NrTow,NrMag,Data,Ilość}

RRK jako język zapytań – przykład (c.d.)

Interpretacja nazw tabel DOSTAWCA^I, DOSTAWA^I

DOSTAWCA

NrDcy	Nazwa	Adres
10	X	A
11	Y	A
12	Z	B
13	W	C

DOSTAWA

NrDcy	NrTow	NrMag	Data	Ilość
10	100	200	a	15
10	150	200	a	20
12	100	201	b	30
11	150	202	c	10

RRK jako język zapytań – przykład (c.d.)

DOSTAWCA

NrDcy	Nazwa	Adres
10	X	A
11	Y	A
12	Z	B
13	W	C

d

DOSTAWA

NrDcy	NrTow	NrMag	Data	Ilość
10	100	200	a	15
10	150	200	a	20
12	100	201	b	30
11	150	202	c	10

w

1. "Podaj nazwy dostawców o adresie B, którzy dostarczali towary o numerze 100 w ilości większej niż 10"

{[Nazwa : *d*.Nazwa] | DOSTAWCA(*d*) $\wedge$ *d*.Adres = 'B' $\wedge$

$\wedge \exists w$ (DOSTAWA(*w*) $\wedge$ *w*.NrTow = 100 $\wedge$ *w*.Ilość > 10 $\wedge$

$\wedge$ *d*.NrDcy = *w*.NrDcy)}.

Zapytanie w RRK – przykład 2

Egzamin

NrStud	NrPrzed
1	a
2	a
3	a
1	b
2	b

Przedmiot

NrPrzed
a
b

"Podaj studentów, którzy zdali egzaminy z wszystkich przedmiotów"

{[NrStud:s.NrStud] | Egzamin(s) $\wedge \forall$p (Przedmiot(p) $\Rightarrow$

$\exists$e Egazmin(e) $\wedge$ e.NrPrzed=p.NrPrzed $\wedge$ e.NrStud=s.NrStud)}

{[NrStud:s.NrStud] | Egzamin(s) $\wedge \neg \exists$p (Przedmiot(p) $\wedge$

$\neg \exists$e Egazmin(e) $\wedge$ e.NrPrzed=p.NrPrzed $\wedge$ e.NrStud=s.NrStud)}

RRD – Relacyjny Rachunek Dziedzin

1. Stanowi teoretyczną podstawę języka QBE (Query By Example).
2. Zakłada ustalone uporządkowanie kolumn.
3. Zmienne przebiegają zbiór wartości prostych i mogą występować jako argumenty predykatów (nazw tabel).

RRD – przykład

DOSTAWCA

NrDcy	Nazwa	Adres
10	X	A
11	Y	A
12	Z	B
13	W	C

DOSTAWA

NrDcy	NrTow	NrMag	Data	Ilość
10	100	200	a	15
10	150	200	a	20
12	100	201	b	30
11	150	202	c	10

"Podaj nazwy tych dostawców o adresie B, którzy dostarczali towary o numerze 100 w ilości większej niż 10"

$\{[Dost : x_n] \mid \exists\, x_d, x_a\ DOSTAWCA(x_d, x_n, x_a) \wedge x_a = \text{"B"} \wedge \exists\, x_m, x_c, x_i\ DOSTAWA(x_d, 100, x_m, x_c, x_i) \wedge x_i > 10\}$

RRD – alfabet

1. Symbole specyficzne (pozalogiczne)
   a) stałe proste: $c_1, c_2, \ldots$
   b) nazwy tabel: $R_1, R_2, \ldots$
2. Symbole ogólne (logiczne)
   a) zmienne proste: $x_1, x_2, \ldots$
   b) operatory porównań: $=, \neq, <, \leq, >, \geq$
   c) symbole logiczne: $\neg, \vee, \wedge, \Rightarrow, \exists, \forall$
   d) ograniczniki: , | { } ( )

Z każdą nazwą relacji R związany jest jej *krotność* (ang. *arity*), tj. liczba wskazująca ilu członowa (argumentowa) jest ta nazwa arity(R) = n, $n \geq 1$.

RRD – reguły tworzenia wyrażeń

1. **Termy**:
   $a ::= c \mid x$
2. **Formuły**:
   $\varphi ::= R(a_1, \ldots, a_n) \mid a_1\ \theta\ a_2 \mid (\varphi) \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid \exists v\, \varphi \mid \forall v\, \varphi$,
   (v musi być zmienną wolną w φ).
3. **Zapytania**:
   $\{[A_1 : a_1, \ldots, A_n : a_n] \mid \varphi\}$,
   (zbiór zmiennych występujących w *krotce wynikowej* $[A_1 : a_1, \ldots, A_n : a_n]$ musi być równy zbiorowi zmiennych wolnych w *kwalifikatorze* φ).

Semantyka – dziedzina interpretacji

Dziedzina interpretacj dla RRK (zbiór atrubutów nieskończony):

$$DZ_{RRD} = (Val \cup Tab)$$

- Val – zbiór wartości proste,
- Tab – zbiór tabel.

Semantyka RRD – interpretacja symboli pozalogicznych alfabetu i wartościowanie zmiennych

$$DZ_{RRK} = (Val \cup Tab)$$

Interpretacja symboli pozalogicznych – funkcja I:

- $c^I \in Val$ – stałym prostym odpowiadają wartości proste,
- $R^I \in Tab$ – nazwom tabel przypisywane są tabele (tej samej krotności co nazwa tabeli).

Wartościowanie zmiennych – funkcja ω:

$$\omega : Var \rightarrow Val$$

RRD – spełnianie formuł

Spełnianie formuł (obliczanie wartości logicznej formuł),
$(I, \omega) \models \varphi$ – spełnianie formuły φ przy interpretacji I i wartościowaniu ω:

$(I, \omega) \models R(a_1, ..., a_n) \Leftrightarrow (a_1, ..., a_n)^I(\omega) \in R^I$,
$(I, \omega) \models a\ \theta\ a' \Leftrightarrow a^I(\omega)\ \theta\ a'^I(\omega)$,
$(I, \omega) \models (\varphi) \Leftrightarrow (I, \omega) \models \varphi$,
$(I, \omega) \models (\varphi \vee \varphi') \Leftrightarrow (I, \omega) \models \varphi \vee (I, \omega) \models \varphi'$,
$(I, \omega) \models (\varphi \wedge \varphi') \Leftrightarrow (I, \omega) \models \varphi \wedge (I, \omega) \models \varphi'$,
$(I, \omega) \models (\exists v\ \varphi) \Leftrightarrow$ istnieje taka stała c, że $(I, \omega[v \rightarrow c]) \models \varphi$,
$(I, \omega) \models (\forall v\ \varphi) \Leftrightarrow$ dla każdej stałej c, $(I, \omega[\ v \rightarrow c\ ]) \models \varphi$.

RRD – odpowiedź na zapytanie

1. $\{t \mid \varphi\}$ – zapytanie wyrażone w RRD.
2. Odpowiedzią na zapytanie $\{t \mid \varphi\}$ nazywamy każdą krotkę $\omega(t)$, określoną przez wartościowanie ω, dla którego spełniony jest kwalifikator φ zapytania, tj. $(I, \omega) \models \varphi$. Szukanie odpowiedzi jest więc równoważne szukaniu właściwych wartościowań zmiennych.
3. Zbiór wszystkich odpowiedzi na zapytanie $\{t \mid \varphi\}$:

$$\{t \mid \varphi\}^I = \{\omega(t) \mid (I, \omega) \models \varphi\}.$$

W praktyce:

R^I – tabela w bazie danych;

$\omega(v)$ – wybór krotki z tabeli i przypisanie jej zmiennej v.

RRD jako język zapytań - przykład

Niech dane będą typy relacji:

DOSTAWCA	– 3-argumentowa,
TOWAR	– 3-argumentowa,
MAGAZYN	– 2-argumentowa,
DOSTAWA	– 3-argumentowa

Kolumny tabel mają więc ustalone i istotne uporządkowanie.

RRD jako język zapytań – przykład

DOSTAWCA

NrDcy	Nazwa	Adres
10	X	A
11	Y	A
12	Z	B
13	W	C

DOSTAWA

NrDcy	NrTow	NrMag	Data	Ilość
10	100	200	a	15
10	150	200	a	20
12	100	201	b	30
11	150	202	c	10

"Podaj nazwy tych dostawców o adresie B, którzy dostarczali towary o numerze 100 w ilości większej niż 10"

{[Dost:*naz*] | ∃*nrd,adr* DOSTAWCA(*nrd, naz, adr*) ∧ *adr = 'B'* ∧
∃ *mag, data, ilość* DOSTAWA(*nrd, 100, mag, data, ilość*) ∧ *ilość* > 10}

Zapytanie w RRD – przykład 2

Egzamin

NrStud	NrPrzed
1	a
2	a
3	a
1	b
2	b

Przedmiot

NrPrzed
a
b

"Podaj studentów, którzy zdali egzaminy z wszystkich przedmiotów"

{[NrStud : s] | ∃p' Egzamin(s,p') ∧ ∀p (Przedmiot(p) ⇒ Egzamin(s,p)}

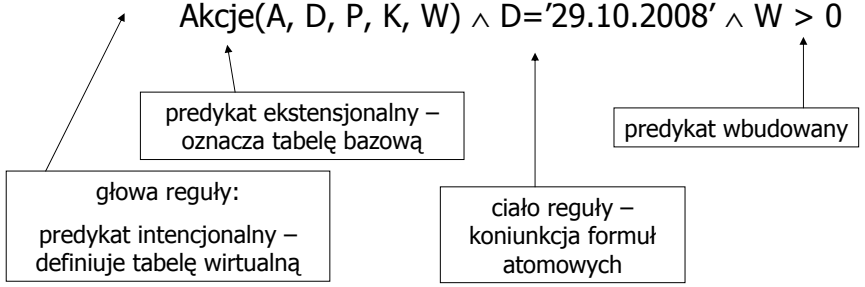
{[NrStud : s] | ∃p' Egzamin(s,p') ∧ ¬∃p (Przedmiot(p) ∧ ¬ Egazmin(s,p)}

Datalog

1. Język operowania w bazie danych wzorowany na Prologu.
2. Różnica:
   - obliczanie programów w Datalogu odbywa się poprzez przekształcenie do wyrażeń algebry relacji operujących na bazie danych - *podejście teoriomodelowe*,
   - Prolog stosuje podejście *teoriodowodowe* (teoria rezolucji)

Zapytania w Datalogu – ogólna postać

- Schemat:
  Akcje(IdAkcji, Data, KursPocz, KursKońc, Zmiana)
- Reguła w Datalogu:
  Wzrosty(A, W) :-
  Akcje(A, D, P, K, W) ∧ D='29.10.2008' ∧ W > 0



Datalog a RRD

1. W Datalogu i w RRD zmienne przyjmują wartości proste.
2. W Datalogu definiowane są ***reguły***, a ciągi reguł tworzą ***programy***. Programy mogą być ***rekurencyjne***.
3. RRD jest językiem pierwszego rzędu, nie można w nim więc definiować rekursji (przykład formułować zapytań wymagających tworzenia domknięcia tranzytywnego).

Predykaty

1. *Ekstensjonalne* - predykaty, których zakresy (w postaci relacji) zapamiętane są w bazie danych jako tabele bazowe, np: **Student**, **Egzamin**
2. *Intencjonalne* - odpowiadające im relacje zdefiniowane są za pomocą wyrażeń (tabele wirtualne) np. **StudenciInformatyki**.
3. *Wbudowane* (*built-in predicates*) – standardowe operatory porównań: =, <, >, >=, <=, ≠

Formuły atomowe

$p(a_1, ..., a_n)$ – *formuła atomowa*,

p – predykat (ekstensjonalny, intencjonalny lub wbudowany),

a_i - zmienna lub stała,

Formuła atomowa wyznacza relację złożoną z tych krotek, dla których formuła jest prawdziwa. W ogólności relacja ta może być skończona lub nieskończona. Formuły zawierające predykaty wbudowane mogą wyznaczać relacje nieskończone.

X < *Y* – wyznacza relację nieskończoną,

X = 100 – wyznacza relację skończoną,

Student(*X*, *Y*, 'Inf') – wyznacza relację skończoną.

Wyrażenia Datalogu

Termy:

a ::= c | X

Formuły:

$\varphi ::= R(a_1,...,a_n) \mid \neg R(a_1,...,a_n) \mid a_1\ \theta\ a_2 \mid \varphi_1 \wedge \varphi_2$

Reguły:

$R(a_1,...,a_n)$:- φ

Programy (zapytania):

ciągi reguł.

Ograniczony zakres zmiennych

Aby każda formuła wyznaczała relację skończoną, to każda występująca w niej zmienna musi mieć **ograniczony zakres**.

Zmienna X ma ograniczony zakres, jeśli:

- występuje w formule atomowej z predykatem ekstensjonalnym lub intencjonalnym,
- jest porównywana symbolem równości '=' ze stałą lub ze zmienną o ograniczonym zakresie.

Mówimy wówczas, że zmienna jest *bezpieczna* (ang. *safe*)

Zmienne o ograniczonych zakresach

Przykłady

Każda zmienna w poniższej formule ma ograniczony zakres

Akcje(*N, D, X, Y*) $\wedge$ *X* < *Y*,

(dla relacji Akcje(Nazwa, Data, KursPocz, KursKońc).

Ograniczone zakresy mają także zmienne w formule:

X = 'WBK' $\wedge$ *Y* = 10.20

Nieograniczony zakres zmienna *Y* w formule:

X = 'WBK' $\wedge$ *Y* $\neq$ 10.20

A co dla *X* i *Y* w formule?

Akcje(*N, D, X, Y*) $\wedge$ *X* = 'WBK' $\wedge$ *Y* $\neq$ 10.20

Reguły – przykład

DOSTAWCA

10	X	A
11	Y	A
12	Z	B
13	W	C

DOSTAWA

10	100	200	a	15
10	150	200	a	20
12	100	201	b	30
11	150	202	c	10

"Podaj nazwy tych dostawców o adresie B, którzy dostarczali towary o numerze 100 w ilości większej niż 10"

WYNIK(*N*) :- DOSTAWCA(*D*, *N*, *A*) $\wedge$ *A* = 'B' $\wedge$
DOSTAWA(*D*,100, _, _, *I)* $\wedge$ *I* > 10

Uwaga: stosujemy symbol podkreślenia '_' na oznaczenie nieistotnej zmiennej

Programy

Programem w Datalogu jest ciąg reguł w Datalogu.

Program jest konieczny dla wyznaczenia sumy dwóch relacji.

Wówczas ta sama głowa (prawa strona) występuje w kilku regułach:

Przykład (studenci z kierunków matematyka i fizyka):

StudMatFiz(IdSt, Nazw) := Student(IdSt, Nazw, K) $\wedge$ K='Mat'

StudMatFiz(IdSt, Nazw) := Student(IdSt, Nazw, K) $\wedge$ K='Fiz'

Zapytanie Datalogu z negacją

Egzamin

NrStud	NrPrzed
1	a
2	a
3	a
1	b
2	b

Przedmiot

NrPrzed
a
b

"Podaj studentów, którzy zdali egzaminy z wszystkich przedmiotów"

R(S) :- Egzamin(S, P') $\wedge$ Przedmiot(P) $\wedge \neg$ Egzamin(S, P)

Q(S) :- Egzamin(S,P) $\wedge \neg$ R(S)

R(S) – studenci, dla których istnieje przedmiot, którego nie zdawali

Q(S) – studenci, którzy nie należą do R(S)

Programy rekurencyjne

Programem rekurencyjnym nazywamy program zawierający regułę rekurencyjną.

Regułą rekurencyjną nazywamy regułę, w której predykat występujący w głowie reguły występuje również w jej ciele.

Możliwość definiowania programów rekurencyjnych znacznie zwiększa **siłę ekspresji** języka. Pozwala między innymi na definiowanie domknięcia tranzytywnego relacji.

Programy rekurencyjne - przykład

Niech Rodzic(D, R) oznacza relację, taką że:

$(d, r) \in$ Rodzic

wtedy i tylko wtedy, gdy osoba *d* ma rodzica (ojca lub matkę) *r*.
Rodzeństwem są osoby, które mają tego samego rodzica.
Kuzynami nazywamy osoby, których rodzice są rodzeństwem lub kuzynami (rekurencja!).
Zadanie: Napisz program w Datalogu definiujący relację Kuzyn(*X,Y*), gdzie *X* i *Y* są kuzynami.

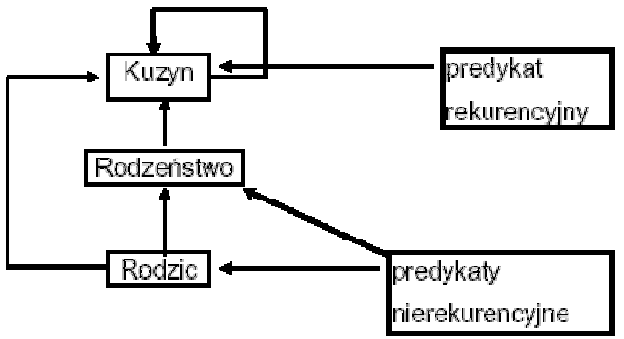
Rodzeństwo(*X, Y*) :– Rodzic(*X, Z*) $\wedge$ Rodzic(*Y, Z*) $\wedge$ $X \neq Y$

Kuzyn(*X, Y*) :– Rodzic(*X, Z*) $\wedge$ Rodzic(*Y, V*) $\wedge$ Rodzeństwo(*Z, V*)

Kuzyn(*X, Y*) :– Rodzic(*X, Z*) $\wedge$ Rodzic(*Y, V*) $\wedge$ Kuzyn(*Z, V*)

Programy rekurencyjne – graf predykatów

Graf zależności predykatów w programie:



Obliczanie programów nierekurencyjnych

Rodzeństwo(X, Y) :– Rodzic(X, Z) $\wedge$ Rodzic(Y, Z) $\wedge$ $X \neq Y$

Przekształcenie na wyrażenia algebry relacji:

R1(X, Y, Z) := Rodzic(X, Z) $\bowtie$ Rodzic(Y, Z)

R2(X, Y, Z) := $\sigma_{X \neq Y}$(R1(X, Y, Z)

Rodzeństwo(X, Y) := $\pi_{X,Y}$(R2(X, Y, Z))

Inaczej:

Rodzeństwo(X, Y) := $\pi_{X,Y}(\sigma_{X \neq Y}$(Rodzic($X$, Z) $\bowtie$ Rodzic(Y, Z)))

Obliczanie programów nierekurencyjnych

Rodzeństwo(X, Y) :– Rodzic(X, Z) $\wedge$ Rodzic(Y, Z) $\wedge$ $X \neq Y$

Rodzeństwo(X, Y) := $\pi_{X,Y}(\sigma_{X \neq Y}$(Rodzic($X$, Z) $\bowtie$ Rodzic(Y, Z)))

Ogólnie:

- zmienne traktowane są jako nazwy atrybutów w relacjach,
- koniunkcja zastępowana jest złączeniem naturalnym,
- predykaty wbudowane zastępowane są selekcję,
- zmienne w głowie reguły wyznaczają projekcję.

Obliczanie programów rekurencyjnych (bez negacji)

Rodzeństwo(X, Y) :– Rodzic(X, Z) $\wedge$ Rodzic(Y, Z) $\wedge$ X $\neq$ Y
Kuzyn(X, Y) :– Rodzic(X, Z) $\wedge$ Rodzic(Y, V) $\wedge$ Rodzeństwo(Z, V)
Kuzyn(X, Y) :– Rodzic(X, Z) $\wedge$ Rodzic(Y, V) $\wedge$ Kuzyn(Z, V)

Rodzeństwo(X, Y) := $\pi_{X,Y}(\sigma_{X \neq Y}$(Rodzic($X$, Z) $\triangleright\!\triangleleft$ Rodzic(Y, Z)))

Kuzyn(X,Y) := $\pi_{X,Y}$(Rodzic(X, Z) $\bowtie$ Rodzic(Y, V) $\bowtie$ Rodzeństwo(Z, V)))

$\cup$ $\pi_{X,Y}$(Rodzic(X,Z) $\bowtie$ Rodzic(Y,V) $\bowtie$ Kuzyn(Z,V)))

Obliczanie reguł rekurencyjnych

Równanie rekurencyjne:

Kuzyn(X,Y) := $\pi_{X,Y}$(Rodzic(X,Z) $\bowtie$ Rodzic(Y,V) $\bowtie$ Rodzeństwo(Z,V)))

$\cup$ $\pi_{X,Y}$(Rodzic(X,Z) $\bowtie$ Rodzic(Y,V) $\bowtie$ Kuzyn(Z,V)))

Iteracyjny proces wyznaczania rozwiązania:

$Kuzyn_0(X,Y)$:= $\pi_{X,Y}$(Rodzic(X, Z) $\bowtie$ Rodzic(Y, Z) $\bowtie$ Rodzeństwo(Z, V)))

$Kuzyn_{i+1}(X,Y)$:= $\pi_{X,Y}$(Rodzic(X,Z) $\bowtie$ Rodzic(Y,V) $\bowtie$ $Kuzyn_i$(Z,V))), $i > 0$.

Rozwiązanie (punkt stały):

Kuzyn(X,Y) = $Kuzyn_0(X,Y) \cup \ldots \cup Kuzyn_N(X,Y)$, dla N takiego, że

$Kuzyn_{N+1}(X,Y) = \{\}$.

Wówczas relację Kuzyn(X,Y) nazywamy *punktem stałym równania*.

Problemy:

1. Czy taki punkt stały istnieje?
2. Czy jest on jednoznaczny (czy najmniejszy)?

Warunki istnienie rozwiązania programów rekurencyjnych

Rozwiązanie istnieje, gdyż:

1. Relacje są skończone.
2. Operacje algebry relacji są monotoniczne.

Twierdzenie: Operacje sumy, projekcji, selekcji i iloczynu kartezjańskiego są monotoniczne.

Suma: $R_1 \subseteq R_2$ i $S_1 \subseteq S_2$, to $R_1 \cup S_1 \subseteq R_2 \cup S_2$.

Projekcja: $R_1 \subseteq R_2$, to $\pi_X(R_1) \subseteq \pi_X(R_2)$.

Selekcja: $R_1 \subseteq R_2$, to $\sigma_E(R_1) \subseteq \sigma_E(R_2)$.

Iloczyn kart: $R_1 \subseteq R_2$ i $S_1 \subseteq S_2$, to $R_1 \ {}_M\times_N \ S_1 \subseteq R_2 \ {}_M\times_N \ S_2$.

Warunki istnienie rozwiązania programów rekurencyjnych (c.d.)

Uwaga: Różnica nie jest monotoniczna !

Różnica: $R_1 \subseteq R_2$ i $S_1 \subseteq S_2$, to $R_1 - S_1 \subseteq R_2 - S_2$ (??).

Stąd problemy, gdy w ciele reguły wystąpią zanegowane formuły.

Podsumowanie

Twierdzenie. Następujące cztery języki są parami równoważne (definiują tę samą klasę funkcji na relacjach w relacyjnej bazie danych):

1.1. Algebra relacji.
2.2. Relacyjny rachunek krotek (RRK)
3.3. Relacyjny rachunek dziedzin (RRD)
4.4. Nierekurencyjny Datalog z negacją.