Mágia čara zbavená

Generalized magic without magic

Ján Šturc

Jar, 2012

Podstata magickej transformácie

- Magická transformácia je optimalizačná technika, ktorá transformuje datalógový program na iný datalógový program s nasledujúcimi vlastnosťami:

1. Počíta tú istú odpoveď ako pôvodný datalógový program.
2. Keď ho vypočítavame seminaívnou (naívnou) evaluáciou, do výpočtu vchádza tá istá množina faktov (n-tíc) ako pri výpočte zhora nadol (SLD-rezolúciou alebo RGT, či QRGT).
3. Selekcie (väzby argumentov) sa posúvajú od cieľov k pocieľom. Z hlavy pravidla postupne (z ľava do prava) k jednotlivým pocieľom v tele.

Úpravy

- RGG vytvoríme k programu (nerektifikovanému) bez ozdôb
  - Z grafu vynecháme IDB predikáty, okrem cieľa v koreni.
  - Spätné šípky vedieme rovno do pravidlových uzlov pre príslušný IDB predikát
- magických a suplementárnych predikátov
  - Obsahujú všetky zaujimavé premenné.
  - Premenná je zaujímavá, ak sa vyskytuje v hlave, alebo sa bude vyskytovať v nejakom ešte nevyhodnotenom cieli pravidla
- úprava výpočtového algoritmu
  - unifikačný join a antijoin
  - operácia ATOV

Unifikačný join

- Relácie obsahujú termy s premennými
- Úpravou „nested loop“ joinu

  Namiesto testu na rovnosť robíme slabú unifikáciu.
- Do výsledku vkladáme „unifikát“ najvšeobecnejší unifikovaný term.
- Z výslednej relácie odstránime n-tice subsumované inými n-ticami („duplikáty“).
- Použitie algoritmov založených na triedení a hašovaní nie je zrejmé. Treba vymyslieť funkciu zachovávajúcu subsumciu. (Asi sa to dá.)

Antijoin

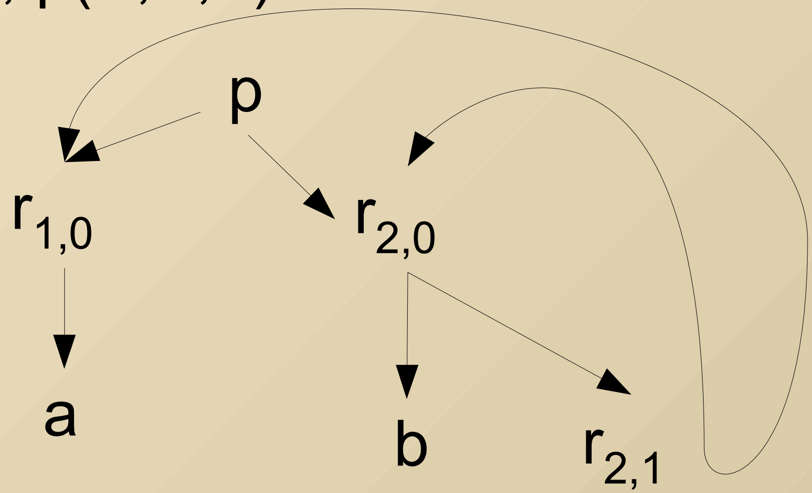
- Nie vždy sa dá v relácii jednoducho reprezentovať napr.
  - *X – f(f(X))*
  - < *X, Y* > - < *X, X* >
- Spoľahlivo funguje rozdiel (antijoin) pre relácie, kde prvá relácia obsahuje len n-tice s uzavretými (ground) termami.
- Matematik si ľahko dokáže predstaviť implementáciu predikátov pomocou postupnosti relácii $\{R_i: 0 \leq i \leq n\}$ a výpočet $R = R_0 - (R_1 - (\ldots - R_n)\ldots))$.
- Otázka, čo je únosná dávka zložitosti závisí od doby a okolností.

Príklad (Ullmann)

r_1: $p(X, Y, W) \leftarrow a(X, Y, W)$.

r_2: $p(X, Y, W) \leftarrow b(W, Y, Z), p(X,X,Z)$.

?- $p(X, Y, 1)$ RGG:



$sup_0(X,Y,1)$.

$sup_0(X,X,Z) \leftarrow sup_{2,1}(X,Y,Z,W)$.

$sup_{2,1}(X,Y,Z,W) \leftarrow sup_0(X,Y,W), b(W,Y,Z)$.

$p(X,Y,W) \leftarrow sup_0(X,Y,W), a(X,Y,W)$.

$p(X,Y,W) \leftarrow sup_{2,1}(X,Y,Z,W), p(X,X, Z)$.

sup_0 zastupuje $sup_{1,0}$ aj $sup_{2,0}$ aj m_p.

Výpočet podľa RGG

1) EDB relácie sú dané.
2) Inicializujeme štartovaciu pomocnú relácia (semeno).
3) Ostatné relácie (intenzionalne a pomocné relácie sú na počiatku prázdne
4) Vetvenie RGG je union (∪) vtoa výrazov pre jednotlivé vetvy.
5) Vnútorný uzol je join/antijoin pomocnej relácie s atov nasledujúceho predikátu.
6) Spätne hrany updatujú nulté pomocné relácie (magické množiny) pomocou lokálnej pomocnej relácie.

Výpočet pomocných relácii.

EDB: $A = \{<m,n,5>\}$
$B = \{<1,2,3>, <3,4,5>\}$

$S_0 = \{<U,V, 1>\}$
$S_{2,1} = \emptyset$
$P = \emptyset$

Program môžeme iterovať v dvoch cykloch: najprv pomocné relácie { S_0 $S_{2,1}$ },potom reláciu P.

Loop { $S_{2,1}(X,Y,Z,W) = S_0(X,Y,W) \bowtie B(W,Y,Z);$
$S_0(X,X,Z) = S_{2,1}(X,Y,Z,W);$ }

$S_{2,1}(X,Y,Z,W)$: $S_0(X,Y,W)$ U,V, 1

U,2,3,1 U,U,3

4,4, 5,3 4, 4 , 5

Vlastný výpočet

Loop

$$\{ P(X, Y, W) = S_0(X,Y,W) \bowtie A(X,Y,W) \cup \Pi_{X,Y,W}(S_{2,1}(X,Y,Z,W) \bowtie p(X,X,Z)); \}$$

Trochu divná notácia pre selekcie a projekcie. Ale, je to s trochou námahy prepísateľné do algebry.

Keďže S_0 a A sa nespájajú výsledok je prázdny.

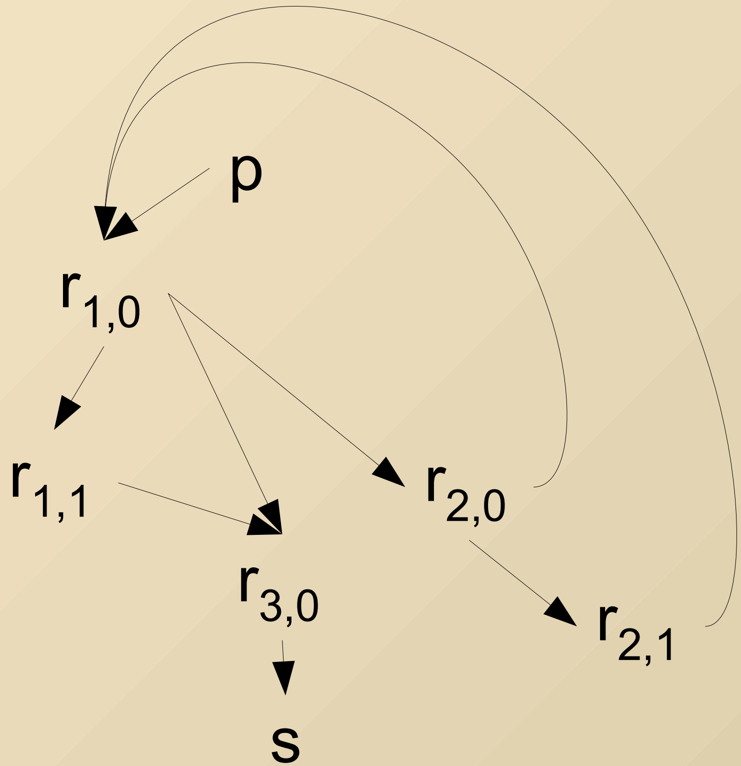
Pozn.: Operácia ATOV pre nerektifikované argumenty, vyžaduje mierne zovšeobecnie.
Striktne vzaté ATOV(p(X,Y), P) je binárna relácia a ATOV(p(X,X), P) unárna relácia. Niekedy potrebujeme, aby aj druhá relácia bola dvojargumentová, treba došpecifikovať mapovanie premenných (atribútov).

Príklad (pracne rektifikovateľný – cvičenie)

r_1: $p(X, Y) \leftarrow q(X, X, Z, Y), q(X, Z, W, a).$

r_2: $q(A, B, C, D) \leftarrow p(A, B), p(C, D).$

r_3: $q(A, B, C, D) \leftarrow s(A, B, C, D).$



Transformovaný program

$sup_{1,0}(X, X, Z, Y) \leftarrow sup_{2,0}(X,Y).$
$sup_{1,0}(X, Z, W, a) \leftarrow sup_{2,1}(X,Y,Z,W).$
$sup_{2,0}(A,B) \leftarrow sup_{1,0}(A, B, C, D).$
$sup_{2,0}(C,D) \leftarrow sup_{1,1}(A, B, C, D).$

$sup_{1,1}(A,B,C,D) \leftarrow sup_{1,0}(A, B, C, D), q(A,A,C,D).$
$sup_{2,1}(A,B,C,D) \leftarrow sup_{2,0}(A,B), p(A,B).$

$p(X,Y) \leftarrow sup_{1,1}(X, Y, Z, W), q(X, Z, W, a).$
$q(A, B, C, D) \leftarrow sup_{2,1}(A,B,C,D), p(C,D).$
$q(A, B, C, D) \leftarrow sup_{2,0}(A,B), s(A,B,C, D).$

semeno je $sup_{2,0}(\_, \_)$.

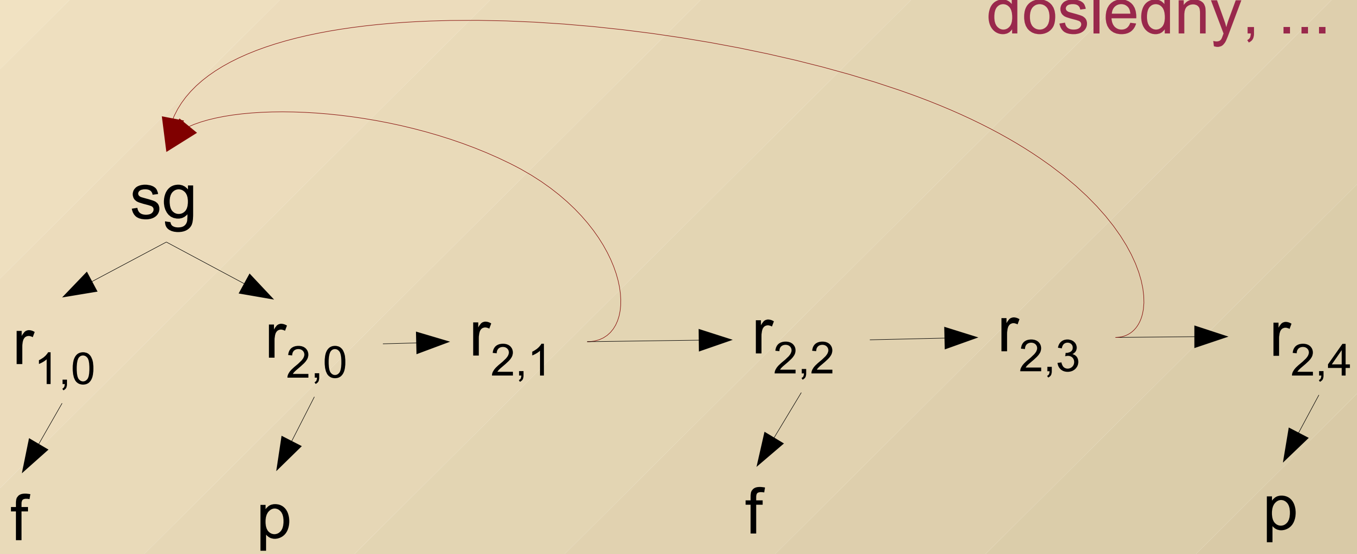
Nelineárna „the same generation“

r_1: $sg(X, Y) \leftarrow f(X, Y).$

r_2: $sg(X, Y) \leftarrow p(X, U), sg(U, V), f(V,W), sg(W, Z), p(Y, Z).$

?- $sg(j, W)$

Keby som bol dôsledný, ...



Po transformácii

$sup_0(U,T) \leftarrow sup_{2,1}(X,Y,U,V).$
$sup_0(W, T) \leftarrow sup_{2,3}(X,Y,W,Z).$

$sup_{2,1}(X,Y,U,V) \leftarrow sup_0(X, Y), p(X, U).$
$sup_{2,2}(X,Y,V,W) \leftarrow sup_{2,1}(X,Y,U,V), sg(U,V).$
$sup_{2,3}(X,Y,W,Z) \leftarrow sup_{2,2}(X,Y,V,W), f(V,W).$
$sup_{2,4}(X,Y,Z) \leftarrow sup_{2,3}(X,Y,W,Z), sg(W,Z).$

$sg(X,Y) \leftarrow sup_0(X, Y), f(X,Y).$
$sg(X,Y) \leftarrow sup_{2,4}(X,Y,Z), p(Z,Y).$

$sup_0(j, T).$

Oranžové premenné sú zbytočné a možno ich vynechať.

Príklad s funkčnými symbolmi

r_1: add(X, 0, X) ← nat(X).

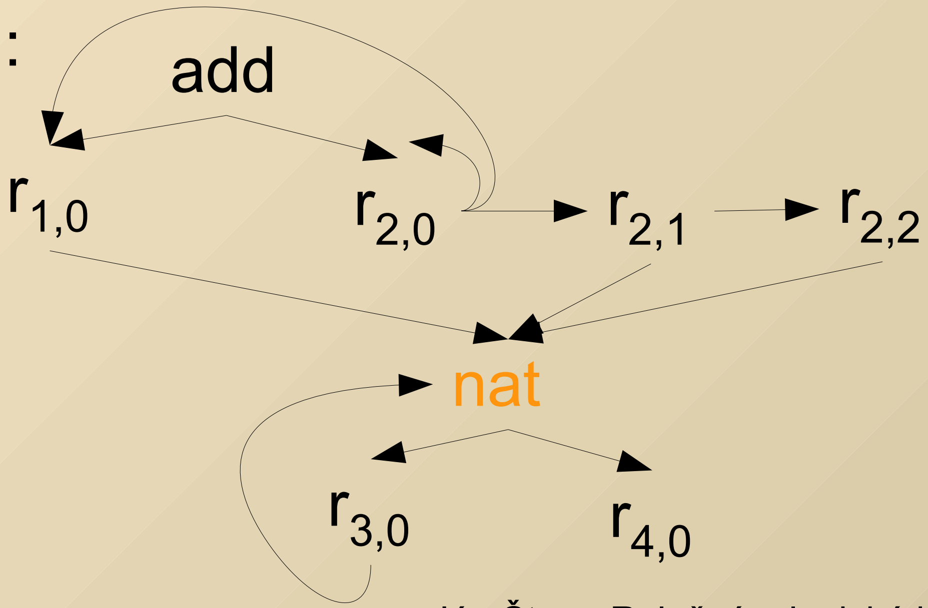
r_2: add(X, s(Y), Z) ← add(s(X),Y, Z), nat(X), nat(Y).

r_3: nat(s(X)) ← nat(X).

r_4: nat(0).

?- add(s(s(0)), s(s(s(0))), Z)

RGG:



Príklad pokračovanie: transformácia

$sup_{a,0}(s(s(0)), s(s(s(0))), Z).$

$sup_{a,0}(s(X), Y, Z) \leftarrow sup_{a,0}(X, s(Y), Z).$

$sup_{n,0}(X) \leftarrow sup_{a,0}(X, \_, \_).$

$sup_{n,0}(Y) \leftarrow sup_{a,0}(\_, Y, \_).$

$sup_{n,0}(X) \leftarrow sup_{n,0}(s(X)).$

$add(X, Y, Z) \leftarrow sup_{a,0}(X, Y, Z), add(X,Y,Z).$

$add(X, 0, X) \leftarrow sup_{a,0}(X, \_, \_), nat(X).$

$nat(s(X)) \leftarrow sup_{n,0}(X), nat(X).$

$nat(0).$

Komentár

- Je to zbytočne komplikované. Prakticky celý výpočet prebieha v $sup_{a,0}$. Predikát nat len kontroluje tvar, výrazov v argumentoch add. Ale výrazy iného tvaru ako s(...s(0)...) ani nemôžu v programe vzniknúť.
- Keď som nahradil pravidlo

  r_2: add(X, s(Y), Z) ← add(s(X),Y, Z), pravidlom

  r_2': add(X, s(Y), s(Z)) ← add(X,Y, Z), sa mi zovšeobecnená magická transformácia nedarila. Resp. som vzniklému programu nerozumel.