EVALUAREA SEMI-NAIVĂ A UNUI SISTEM DE REGULI RELATIV LA O CERERE CE CONȚINE VARIABILE LEGATE

Ing. Elena Manoilă

Institutul de Cercetări în Informatică

Rezumat

Este tratată problema evaluării eficiente a cererilor recursive, exprimate sub forma de clauze Horn. Se definesc formal strategii de transmitere a informațiilor și se arată cum se pot implementa prin rescrierea unui program dat și prin evaluarea sa bottom-up.

O metodă simplă, dar extrem de ineficientă ar fi aceea de a evalua întreaga relație corespunzătoare cererii, după care se aplică operatorul de selecție. Ineficiența rezultă din aceea că, se fac foarte multe deducții irelevante pentru cerere.

Materialul prezintă un algoritm de rescriere a unui sistem de reguli, astfel încît, evaluarea semi-naivă a acestuia să nu mai efectueze deducții irelevante pentru cerere. Algoritmul se bazează pe utilizarea legărilor oferite de cerere.

Cuvinte cheie SIP (Sideway Information Passing graph), ornamentare, mulțimi magice, predicate magice, evaluare semi-naivă.

1. Introducere

Să considerăm următorul sistem de reguli:

anc(x,y):– par(x,y).
anc(x,y):–par(x,z),anc(z,y).

Fie următoarea mulțime de fapte (BDE):

par(a,b).
par(b,c).
par(c,d).
par(e,f).
par(f,g).
par(j,i).

Pentru cererea anc(a,y), evaluarea semi-naivă, urmată de o selecție, implică deducția următoarelor tuple ale relației anc, corespunzătoare cererii:
(a,b), (a,c), (a,d), (b,d), (b,c), (c,d), (e,f), (f,g), (j,i), (e,g).
În urma selecției, doar primele trei tuple se dovedesc utile, restul fiind, prin urmare, irelevante.
Metoda mulțimilor magice, generalizată oferă un algoritm de rescriere a sistemului inițial de reguli, astfel încît, în urma evaluării bottom-up, doar faptele relevante pentru cerere să fie calculate.

2. Strategii de transmitere a informației într-o clauză Horn

O metodă de transmitere a informației este SIP. Fiind date niște legări pentru variabilele unui predicat, rezolvăm predicatul cu aceste legări și, astfel, obținem legări pentru alte variabile ale sale; aceste noi legări se pot pasa altui predicat din regula pentru a restrînge calculul etc.

DEFINIȚIE

a). Două variabile ale unei reguli sînt conectate dacă apar ambele într-un predicat. Această relație se extinde în mod tranzitiv, la conectarea printr-un lanț de variabile, în care fiecare pereche adiacentă e conectată.
b). Două predicate sînt conectate dacă fiecare conține una din variabilele unei perechi conectate: și această relație este tranzitivă.
Se observă că, relația de conectivitate este o relație de echivalență (și pe variabile și pe predicate). Deci, mulțimea aparițiilor de predicate dintr-o regulă este reuniunea componentelor conectate. Una din ele conține capul regulii. Celelalte componente, dacă există, sînt subteluri existențiale, ce se rezolvă independent de legările variabilelor capului. Strategiile de SIP au ca scop o singură componentă conectată. Cum cel mai important e modul de transmitere a informației plecînd de la cap, se presupune, fără a pierde din generalitate, ca ocurentele predicatelor unei reguli formează o singură componentă conectată.

DEFINIȚIE

Fie r o regulă, cu capul $p(\theta)$, θ^b - submulțimea argumentelor lui p (deci a lui θ) care sînt legate. Fie p_h un predicat semnificînd p restricționat la argumentele sale legate. Deci, argumentele lui p_h sînt θ^b. Fie P(r), mulțimea aparițiilor predicatelor din corpul regulii r.
Un SIP pentru r e un graf etichetat ce satisface următoarele condiții:

1) Arcele au forma N-→ q, unde $N \subset P(r) \cup \{p_h\}$ și $q \in P(r)$; arcele sînt etichetate cu λ = mulțimea variabilelor, care în urma evaluării predicatelor din N și a efectuării join-ului lor, devin legate. Aceste variabile apar toate în argumentele lui q.

Formalizat:

(i) fiecare variabilă din λ apare în N
(ii) fiecare predicat din N e conectat cu o variabilă din λ (pentru că el nu are nici o utilitate în legarea vreunei variabile)
(iii) cel puțin un argument al lui q are toate variabilele în λ (pot exista și funcții printre argumente, și atunci acel argument e considerat legat, dacă și numai dacă, toate variabilele pe care le conține sînt legate); mai mult, orice variabilă din λ apare într-un argument al lui q ce satisface această condiție. Prin urmare, toate variabilele din sînt utile în legarea a cel puțin unui argument al lui q.

2). Relația de precedență definită mai jos este aciclică:

- p_h precede toate elementele lui P(r);
- un predicat ce nu apare în graf, urmează orice predicat ce apare în graf;
- dacă N-→ q e un arc și $q \in N$, atunci q' precede q.

Condiția 2) exprimă faptul că închiderea tranzitivă a

acestei relații e o relație de ordine parțială, adică exclude posibilitatea ca două predicate să presupună reciproc că celălalt leagă o anumită variabilă.

Un SIP semnifică, deci, următoarea ordine de evaluare a regulii: orice nod în care nu intră arce e evaluat cu toate argumentele libere (excepție făcînd predicatul special p_h, care poate fi gîndit ca un predicat de bază, ale cărui tuple sînt cele impuse de legări). Un predicat în care intră arce e evaluat doar pentru valorile ce sosesc prin arce. În final, cînd toate predicatele au fost evaluate, ele sînt join-ed și rezultatul e proiectat pe variabilele capului regulii, pentru a fi întors ca rezultat (acest join, ca și evaluarea predicatelor, se poate face în mai mulți pași, pe măsură ce tuplele se generează sau o dată cînd toate tuplele sînt disponibile). Această interceptare a SIP-ului spune ce se face și nu cum se face.

Vom numerota de la 1 ocurentele unui predicat în corpul unei reguli.

EXEMPLU

```
sg(x,y):–flat(x,y).
sg(x,y):-up(x,z1),sg(z1,z2),flat(z2,z3),
sg(z3,z4),down(z4,y).
query:–sg(a,y).
```

Un posibil SIP pentru a doua regulă este:

$\{sg_h\} \xrightarrow{x} up$; $\{sg_h, up\} \xrightarrow{z1} sg.1$;

$\{sg_h, up, sg.1\} \xrightarrow{z2} flat$; $\{sg_h, up, sg.1\} \xrightarrow{z3} sg.2$;

Există arce ce intră, atît în predicate de bază, cît și în predicate derivate. Transmiterea informației e importantă pentru ambele tipuri, dar pentru considerente diferite. Un predicat de bază e mereu direct evaluabil; legarea informației e folosită ca o condiție de selecție. La predicatele derivate, legările influențează calculul prin restrîngerea telurilor generate. Deoarece ne ocupăm de propagarea legărilor pentru a eficientiza evaluarea în prezența recursiei, generalizăm notația pentru a permite reprezentări succinte ale SIP-urilor, în care doar arce care intră în predicate derivate sînt reprezentate.

În loc de o mulțime la capătul arcului, permitem două mulțimi (deci, o pereche ordonată de două mulțimi), a doua conținînd doar predicate de bază. Folosim notația $N_1;N_2 \rightarrow q$ pentru a desemna că predicatele din N_1 sînt evaluate și joncționat, acesta oferind legări utilizate în evaluarea predicatelor de bază din N_2. Apoi, join-ul lui $N_1 \cup N_2$ e utilizat pentru a produce legări ce se transmit pe arc.

În această prezentare, SIP-ul precedent devine:

$\{ sg_h ; up \} \xrightarrow{z1} sg.1$; $\{ sg_h , up , sg.1 ; flat \} \xrightarrow{z3} sg.2$

3. Ornamentarea regulilor

Fie P, un program constînd dintr-o mulțime de clauze Horn. Fie cererea $q(\bar{c}, \bar{x})$, unde $\bar{c}$ este vectorul argumentelor legate, și $\bar{x}$ vectorul argumentelor libere. Construim o versiune nouă, ornamentată a programului, notată P^{ad}. În construcție vom înlocui predicatele derivate cu versiuni ornamentate, fiecare predicat putînd avea mai multe variante ornamentate. Pentru fiecare predicat ornamentat p^a și pentru fiecare regulă cu capul p, alegem un SIP pe care-l folosim apoi pentru a genera o versiune ornamentată a regulii: capul noii reguli va fi p^a. Procesul de ornamentare începe cu predicatul cererii. În general, vom avea o colecție de predicate ornamentate, care se îmbogățește pe parcurs. Dacă p^a e un predicat ornamentat, atunci pentru fiecare regulă cu capul p, generăm o versiune ornamentată a regulii și o adăugăm la P^{ad}; spunem că p a fost procesat și-l marcăm ca atare. Versiunea ornamentată a regulii conține și alte predicate ornamentate suplimentare, și acestea se adaugă la colecție, dacă nu figurau deja. Procesul se încheie cînd nici un predicat ornamentat nemarcat n-a rămas. Terminarea e garantată de finitudinea numărului de versiuni ornamentate ale predicatelor unui program.

Ornamentarea unei reguli de capul p, pentru ornamentul a al predicatului p, se face astfel: capul noii reguli e p_h (p restricționat la argumentele sale legate); pentru fiecare ocurenta p_i a unui predicat din regulă, fie λ_i = reuniunea etichetelor arcelor ce intră în p_i (dacă nu intră arce, λ_i=0). Înlocuim p_i cu $p_i^{a_i}$, unde un argument al lui p_i e legat în a_i dacă toate variabilele ce apar în el sînt și în λ_i (dacă nu intră arce în el, atunci ornamentul conține doar f-uri, deci e ca și cum ar fi neornamentat din punct de vedere al scopurilor noastre). Argumentele predicatelor din noua regulă rămîn aceleași ca-n regula inițială. Întrucît ornamentele atașate predicatelor unei reguli sînt determinate de SIP-ul ales, acesta e atașat regulii.

Pentru exemplul precedent, rescrierea se face:

```
sg^bf(x,y): – flat(x,y).
sg^bf(x,y):-up(x,z1),sg^bf(z1,z2),flat(z2,z3),sg^bf(z3,z4),
            down(z4,y).
query: sg^bf(a,y).
```

4. Algoritmul de rescriere a sistemului de reguli

Mai departe, considerăm doar mulțimea regulilor ornamentate ca mai sus, P^{ad}. Se definesc niște predicate suplimentare, ce calculează valorile pasate de la un predicat la altul în regulile inițiale, potrivit SIP-ului ales pentru fiecare. Fiecare din reguli este apoi modificată, astfel încît să se aplice doar cînd valori pentru acele predicate nou definite sînt disponibile. Noile predicate se numesc predicate magice și mulțimile de valori pe care ele le calculează se numesc mulțimi magice. Intenția este ca evaluarea bottom-up a mulțimii de reguli modificate să simuleze SIP-ul selectat, restrîngînd spațiul de căutare.

Transformările ce se fac în continuare sînt următoarele:

1) Creăm pentru orice predicat p^a din P^{ad} nou predicat magic $\_p^a$: aritatea noului predicat este numărul b-urilor din ornamentul a și argumentele sale corespund argumentelor legate ale lui p^a;

2) Pentru orice regulă r din P^{ad} și pentru fiecare ocurentă a unui predicat p^a în corp, generăm o regulă magică prin care se definește magic $\_p^a$ (dacă predicatul ornamentat are mai multe ocurente în corpul unei reguli, mai multe reguli ce definesc p^a se vor genera);
3) Orice regulă e modificată prin adăugarea predicatelor magice în corpul său;
4) Creăm un fapt obținut din cerere, care oferă o valoare inițială pentru predicatul magic, corespunzător predicatului cererii.

Detalierea pasului 2

Folosim litere grecești pentru a desemna liste de argumente. Dacă λ este lista de argumente a predicatului p^a, atunci $\lambda^f(\lambda^b)$ reprezintă λ, cu toate argumentele bound(free) șterse. Considerăm următoarea regulă:

$r: p^a(\lambda) :- q_1^{a_1}(\theta_1), q_2^{a_2}(\theta_2), \ldots q_n^{a_n}(\theta_n)$

Fie s_r SIP-ul asociat regulii. Presupunem că am ordonat predicatele din corpul regulii potrivit SIP-ului (deci, potrivit ordinii induse de condiția 3 din definiția SIP-ului).

a) Fie q_i și $N \rightarrow q_i$ unicul arc ce intră în q_i în SIP. Generăm o regulă magică astfel: capul regulii este $magic\_q_i^{a_i}(\theta_i^b)$. Dacă q_j, $j<i$ este în N, adăugăm $q_j^a{}_j(\theta_j)$ corpul regulii magice. Dacă q_j este un predicat ornamentat și ornamentul conține cel puțin un b, adăugăm și magic $q_j^a{}_j(\theta_j)$ în corp. Dacă predicatul special p_h e în N, adăugăm $magic\_p^a(\lambda^b)$ corpului regulii magice.

b) Fie q_i astfel încît există mai multe arce care intră în el. Atunci, pentru orice arc $N_j \rightarrow q_i$, cu eticheta λ_j, definim o regulă cu capul: head $-q_i - j(\lambda_j)$. Corpul regulii e la fel ca și-n cazul regulii magice de la punctul a. Apoi, regula magică are capul $magic\_q_i^{a_i}(\theta_i^b)$ și corpul conține head $-q_i - j(\lambda_j)$, pentru orice j.

Detalierea pasului 3

Modificăm regula originală, inserînd ocurentele predicatelor magice, corespunzătoare predicatelor derivate din corp și predicatului din cap, în corpul regulii. Pozițiile de inserare sînt: pentru $magic\_p^a$, la stînga corpului regulii, înaintea tuturor celorlalte predicate; pentru $magic\_q_i^{a_i}$, chiar înainte de $q_i^{a_i}$, (intuitiv, $magic\_q_i^{a_i}$, calculează valorile ce se pot pasa lui $q_i^{a_i}$ din stînga, în timpul evaluării regulii; inserarea servește ca o gardă).

Pe același exemplu, în urma transformărilor descrise mai sus, obținem:

$magic\_^{bf}sg(a)$.

r1: $magic\_sg^{bf}(z1) :- magic\_sg^{bf}(x), up(x,z1)$.

r2: $magic\_sg^{bf}(z3) :- magic\_sg^{bf}(x), up(x,z1), sg^{bf}(z1,z2), flat(z2,z3)$.

De fapt, ar fi apărut și $magic\_sg^{bf}(z1)$, dar cum el este implicat logic de precedentele două, în virtutea regulii r1, nu-l mai scriem:

r3: $sg^{bf}(x,y) :- magic\_sg^{bf}(x), flat(x,y)$.

r4: $sg^{bf}(x,y) :- magic\_sg^{bf}(x), up(x,z1), sg^{bf}(z1,z2), flat(z2,z3), sg^{bf}(z3,z4), down(z4,y)$.

Și aici ar fi apărut magic_sg (z1) și magic_sg (z3), dar, la fel ca mai sus, ele sînt consecințe logice ale celorlalte predicate.

În [2] este demonstrată corectitudinea transformărilor, deci, echivalenta programului original, relativ la cererea neornamentată cu programul ornamentat, relativ la cererea ornamentată.

Să vedem cum diferă evaluarea pe următoarea bază de date extensională:

flat(a,b).
flat(e,f).
flat(d,e).
flat(f,h).
up(c,d).
down(h,g).

Fie cererea sg(a,y).

Versiunea netransformată produce în urma evaluării semi-naive:

Pasul 1
sg=∅
d_sg={(e,f),(d,e),(f,h),(a,b)}
Pasul 2
sg={(e,f),(d,e),(f,h),(a,b)}
d_sg={(c,g)}
Pasul 3
sg={(e,f),(d,e),(f,h),(a,b),(c,g)}
d_sg=∅

Cum nici un tuplu nou n-a mai fost obținut, calculul se încheie.

Din toată relația sg, doar tuplul (a,b) este relevant pentru cerere.

Versiunea ornamentată a programului:

r1: $magic\_sg^{bf}(z1) :- magic\_sg^{bf}(x), up(x,z1)$.

r2: $magic\_sg^{bf}(z3) :- magic\_sg^{bf}(x), up(x,z1), sg^{bf}(z1,z2), flat(z2,z3)$.

r3: $sg^{bf}(x,y) :- magic\_sg^{bf}(x), flat(x,y)$.

r4: $sg^{bf}(x,y) :- magic\_sg^{bf}(x), up(x,z1), sg^{bf}(z1,z2), flat(z2,z3), sg^{bf}(z3,z4), down(z4,y)$.

$magic\_sg^{bf}(a)$.

Pasul 1
$sg^{bf}=\emptyset$
$d\_sg^{bf}=\{(a,b)\}$
$magic\_sg^{bf}=\emptyset$
$d\_magic\_sg^{bf}=\{a\}$
Pasul 2
$sg^{bf}=\{(a,b)\}$
$d\_sg^{bf}=\emptyset$
$magic\_sg^{bf}=\{a\}$
$d\_magic\_sg^{bf}=\emptyset$

Cum nici un tuplu nou n-a mai fost dedus la acest pas, procesul se încheie.

Se observă că varianta neornamentată a efectuat foarte

multe deducţii inutile pentru răspuns.

BIBLIOGRAFIE

1. BANCILHON, FR.: Magic Sets and Other Strange Ways to Implement Logic Programs, 1986, (XC), MCC Internal Report, Austin, Texas.
2. RAMAKRISNAN, R.: On the Power of Magic. MCC Internal Report, Austin, Texas (XC), 1988.
3. ZANIOLO, C.: Rule Rewriting Methods for Efficient Implementations of Horn Logic. MCC Internal Report, 1987, Austin, Texas, (XC).
4. BANCILHON, FR.: Naive Evaluation of Recursively Defined Relations. Proc. of the Islamadora Workshop on Databases & AI, Editura Springer, Berlin, 1986, pp.123-140.
5. ZANIOLO, C.: Optimization in a Logic Based Language for Knowledge and Data Intensive Applications. MCC Internal Report, Austin, Texas, 1988, (XC).