XSB

Pallinger Péter

2004. november 23.

Válogatott Fejezetek a Logikai Programozásból

1. A logikai programozás problémái

- A prolog nem deklaratív
  (var/1, assert/1, !/0, ...és még valami...)
- Az SLD rezolúció minden ága:
  - sikerül,
  - meghiúsul,
  - *soha nem ér véget.*

- Nehéz korrekt (minden esetben megálló) programot írni.
- Már egyszerű esetekben is előfordulhat végtelen hurok:

```
tca(X,Y):-a(X,Y).
tca(X,Y):-a(X,Z),tca(Z,Y).
a(1,1). a(2,1).
:- tca(1,2).
```

(Ez az algoritmus csak DAG-ra ad helyes eredményt.)

- Programozásnál elkerülhetjük a ciklikus gráfokat, de adatbázisoknál...?
- Fixpont szemantikát érdemes használni: OLDT rezolúció

2. OLDT rezolúció

- SLD-rezolúcióhoz hasonló
- Elkerüli a rednudáns számítást: memoization/ tabling/ lemmatization/ stb.
- Megáll minden kérdésre, aminek véges a *minimális* modellje.

2.1. prolog példa #1

```
avoids(X,Y):- owes(X,Y).
avoids(X,Y):- owes(X,Z), avoids(Z,Y).
owes(andy,bill).
owes(bill,carl).
owes(carl,bill).
```


2.2. prolog példa #2

```
avoids(X,Y):-avoids(X,Y,[]).
avoids(X,Y,L):-owes(X,Y), \+ member(Y,L).
avoids(X,Y,L):-owes(X,Z), \+ member(Z,L),
               avoids(Z,Y,[Z|L]).
owes(andy,bill).    owes(andy,bob).
owes(bill,carl).    owes(bob,carl).
owes(carl,dan).     owes(carl,dave).
owes(dan,evan).     owes(dave,evan).
owes(evan,fred).    owes(evan,frank).
owes(fred,george).  owes(frank,george).
```


2.3. XSB példa #1

```
:- table avoids/2.
avoids(X,Y):- owes(X,Y).
avoids(X,Y):- owes(X,Z), avoids(Z,Y).
owes(andy,bill).
owes(bill,carl).
owes(carl,bill).
```

3. Az XSB következtetője

3.1. SLG rezolúció

- A célokat ún. "eredménytároló cél-szerver"-nek küldi, minden célhoz egy külön szerver jön létre, ha még nincs ilyen.
- Minden cél-szerverhez külön levezetési fa tartozik.
- Az alábbi három szabály írja le a szerverek működését:
  - Program klóz rezolúció (Program Clause Resolution)
  - Részcél hívás (Subgoal call)
  - Válasz klóz rezolúció (Answer Clause Resolution)

3.2. Alapötlet

- SLG rezolúció használata alapértelmezett stratégiaként
- SLDNF használata optimalizációra

3.3. XSB példa #2

```
:- table avoids/2.
avoids(X,Y):-owes(X,Y).
avoids(X,Y):-avoids(X,Z), owes(Z,Y).
owes(1,2).
owes(2,3).
...
owes(99,100).
owes(100,1).
```

- Jobbrekurzív esetben: $O(n^2)$
- Balrekurzív esetben: $O(n)$

3.4. XSB példa #3

```
sameSCC(X,Y):-reach(X,Y),reach(Y,Z).
:- table reach/2.
reach(X,X).
reach(X,Y):-reach(X,Z),edge(Z,Y).
```

- Időigény: $O(ne)$

3.5. XSB példa #4

```
sameSCC(X,Y):-reachfor(X,Y),reachback(X,Y).
:- table reachfor/2, reachback/2.
reachfor(X,X).
reachfor(X,Y):-reachfor(X,Z),edge(Z,Y).
reachback(X,X).
reachback(X,Y):-reachback(X,Z),edge(Y,Z).
```

- Időigény: $O(e)$

3.6. Automatikus eredménytárolás

- `:- auto_table.`
  - Minimális predikátumhalmazt keres úgy, hogy a köröket lefedje (predikátumok számában exponenciális).
  - Datalog program (amiben nincs rekurzív adastruktúra) mindig meg fog állni.
- `:- suppl_table(+Integer)`
  - Adatbázis-kezelés gyorsítása.

4. Adatbázis-kezelés

```
%student(StdId, StdName, Yr).
%enroll(StdId, CrsId).
%course(CrsId, CrsName).
yrCourse(Yr,CrsName):-
    student(StdId, _, Yr),
    enroll(StdId, CrsId),
    course(CrsId, CrsName).
```

- Az utolsó hívás minden hallgatóra lefut!

4.1. Adatbázis-kezelés gyorsítása

```
yrCourse(Yr,CrsName):-
    yrCrsId(Yr, CrsId),
    course(CrsId, CrsName).
:- table yrCrsId/2.
yrCrsId(Yr,CrsId):-
    student(StdId, _, Yr),
    enroll(StdId, CrsId).
```

- Az XSB ezt a lépést automatikusan is képes elvégezni.

```
:- edb student/3, enroll/2, course/2.
:- suppl_table(2).
yrCourse(Yr,CrsName):-
    student(StdId, _, Yr),
    enroll(StdId, CrsId),
    course(CrsId, CrsName).
```


4.2. Datalog programok max. komplexitása

Ha minden predikátum eredménytárolt:

$\sum_{clause}(len(clause) + k^{num\_of\_vars(body(clause))})$

Ahol k a program konstansainak száma

4.3. Max. komplexitás csökkentése

- ```
  :- p(A,B,C,D), q(B,F,G,A), r(A,C,F,D),
     s(D,G,A,E), t(A,D,F,G).
  ```
  $O(n^7)$
- ```
  :- f1(A,C,D,F,G), r(A,C,F,D), s(D,G,A,E),
     t(A,D,F,G).
  f1(A,C,D,F,G):- p(A,B,C,D), q(B,F,G,A).
  ```
  $O(n^6)$
- Az optimális faktorizáció megtalálása NP-nehéz.

5. Nyelvi elemzés

```
:- table expr/2, term/2.
expr-->expr,[+],term.
expr-->term.
term-->term,[*],primary.
term-->primary.
primary-->['('],expr,[')'].
primary-->[Int],{integer(Int)}.
```

Ez balrekurzív, de eredménytárolással biztosan lefut polinom időben.

5.1. Nyelvtan kiértékeléssel

```
:- table expr/3, term/3.
expr(Val)-->expr(Eval),[+],term(Tval),
            {Val is Eval+Tval}.
expr(Val)-->term(Val).
term(Val)-->term(Tval),[*],factor(Fval),
            {Val is Tval*Fval}.
term(Val)-->factor(Val).
factor(Val)-->primary(Num),[^],factor(Exp),
              {Val is floor(exp(log(Num)*Exp)+0.5)}.
factor(Val)-->primary(Val).
primary(Val)-->['('],expr(Val),[')'].
primary(Int)-->[Int],{integer(Int)}.
```

Itt a jobbrekurzív rész négyzetes lenne eredménytároltan.

5.2. Elemzők max. koplexitása

- $O(n^{k+1})$
  n: a bemenet hossza
  k: a jobboldalak legnagyobb hossza
  (terminális és nemterminális együtt)
- Chomsky normálformában: $O(n^3)$
- Automatikusan:

  ```
  :- suppl_table(2).
  :- edb word/2.
  ```
- A prologban lévő elemző egy „rekurzív mélységi felimerő”.
- Az XSB-ben lévő elemző az Early-algoritmus egy változata.

6. HiLog programozás

```
closure(R)(X,Y) :- R(X,Y).
closure(R)(X,Y) :- R(X,Z),closure(R)(Z,Y).
:- hilog parent.
parent(andy,bob).
parent(bob,cecil).
```


6.1. Map megvalósítása

```
map(F)([],[]).
map(F)([X|Xs],[Y|Ys]):-F(X,Y),map(F)(Xs,Ys).
:- hilog successor,double,square.
successor(X,Y):- Y is X+1.
double(X,Y):- Y is X+X.
square(X,Y):- Y is X*X.
```

7. Egyéb alkalmazások

- automataelmélet
- dinamikus programozás
- aggregátumok számítása
- metaértelmezők készítése

Hivatkozások

[1] The XSB manual, 2003.06.17,

[2] Programming in Tabled Prolog (draft), David S. Warren, 1999.07.21,
`ftp://ftp.cs.sunysb.edu/pub/XSB/doc/XSB/prog_XSB_book_dr.ps.gz`

Köszönöm a figyelmet!