Logikai programozás és szakértői rendszerek (gyakorlat)

Dr. Kusper Gábor

1. gyakorlat

aries.elte.hu/~gruesper/2005_osz/mest-int gyak/

prolog interpreter → visual prolog

.VIP\BIN\WIN\32\VIP.EXE

$4_p > 3_z$ $\quad\quad$ $3_p < 5_e$

2. gyakorlat
(kulapól)

APJA.PRO

```
domains
  ember = symbol

predicates
  nondeterm apja(ember, ember)
  nondeterm unokatestver(ember, ember)
  nondeterm ose(ember, ember)
  nondeterm felesegi(ember, ember)
  nondeterm anyja(ember, ember)

clauses
  apja(adam, peter).
  apja(adam, bela).
  apja(peter, jozsef).
  apja(peter, mate).
  apja(bela, mark).
  apja(bela, istvan).
  felesegi(adam, eva).
  felesegi(peter, judit).
  felesegi(bela, reka).
  unokatestver(Z1, Z2) :-
          apja(Y1, Z1),
          apja(Y2, Z2),
          not(Y2=Y1),
          apja(X, Y1),
          apja(X, Y2).
  ose(X, W)  :- apja(X, W).
  ose(X, W)  :- apja(X, Y), ose(Y, W).
  anyja(Y, X) :- apja(Z, X), felesegi(Z, Y).
  nagyapa(X, Z) :- apja(X, Y), apja(Y, Z).
  nagyi(X, Z)  :- nagyapa(Y, Z), felesegi(Y, X).

GOAL
  anyja(X, mate).
% unokatestver(X,istvan).
```

CSALADFA domains

```
  ember = symbol
predicates
  nondeterm fiu(ember)
  nondeterm lany(ember)
  nondeterm fia(ember, ember)
  nondeterm lanya(ember, ember)
  nondeterm apja(ember, ember)
  nondeterm anyja(ember, ember)
  nondeterm mama(ember, ember)
  nondeterm papa(ember, ember)
clauses
  fiu(adam).
  fiu(zsolt).
  fiu(geza).
  fiu(attila).
  fiu(zoltan).

  lany(eva).
  lany(marta).
  lany(maria).
  lany(anna).
  lany(bea).

  fia(adam, zsolt).
  fia(adam, geza).
  fia(eva, zsolt).
  fia(eva, geza).
  fia(zsolt, attila).
  fia(maria, attila).
  fia(attila, zoltan).
  fia(bea, zoltan).
  lanya(adam, marta).
  lanya(eva, marta).
  lanya(zsolt, anna).
  lanya(maria, anna).
  apja(X, Y) :- fia(Y, X), fiu(Y). % X-nek az apja Y, ha Y fia X-nek
  apja(X, Y) :- lanya(Y, X), fiu(Y).
%  anyja(X, Y):- fia(Y, X), lany(Y).% X-nek az anyja Y, ha Y fia X-nek es Y lany
%  anyja(X, Y):- lanya(Y, X), lany(Y).
  anyja(X, Y):- fia(Y, X), lany(Y) ; lanya(Y, X), lany(Y).
  mama(X, Z) :- apja(X, Y), anyja(Y, Z).
  mama(X, Z) :- anyja(X, Y), anyja(Y, Z).
  papa(X, Z) :- apja(X, Y), apja(Y, Z).
  papa(X, Z) :- anyja(X, Y), apja(Y, Z).
  ose(X, W)  :- apja(X, W).
  ose(X, W)  :- anyja(X, W).
  ose(X, W)  :- apja(X, Y), ose(Y, W).
  ose(X, W)  :- anyja(X, Y), ose(Y, W).
goal
  papa(zoltan, A). PRO
```

CSALADFA2.PRO

```
domains
  ember = symbol
predicates
  nondeterm fiu(ember)
  nondeterm lany(ember)
  nondeterm apja(ember, ember) % apja(X,Y): X apja Y-nak
  felesege(ember, ember) % felesege(X,Y): X felesege Y-nak
  nondeterm anyja(ember, ember) % anyja(X,Y): X anyja Y-nak
  nondeterm fia(ember, ember) % fia(X,Y): X fia Y-nak
  nondeterm lanya(ember, ember) % lanya(X,Y): X lanya Y-nak
  nondeterm nagyi(ember, ember) % nagyi(X,Z): X nagyanyja Z-nek
  nondeterm papi(ember, ember) % papi(X,Z): X nagyapja Z-nek
  nondeterm ose(ember, ember) % ose(X,W): X ose W-nek
clauses
%adatbazis: a tenyek
  fiu(istvan). fiu(peter). fiu(zoltan).
  fiu(zsolt). fiu(bela).
  lany(borbala). lany(eva). lany(katarina).
  lany(anna). lany(antonia). lany(viktoria).
  apja(istvan, peter). apja(istvan, borbala).
  apja(istvan, eva). apja(peter, zoltan).
  apja(peter, zsolt). apja(peter, katarina).
  apja(zoltan, bela).
  felesege(anna, istvan).
  felesege(antonia, peter).
  felesege(viktoria, zoltan).
%szabalyok
  anyja(X, Y) :- apja(Z, Y), felesege(X, Z).
  fia(X, Y) :- apja(Y, X), fiu(X).
  fia(X, Y) :- anyja(Y, X), fiu(X).
% fia(X, Y) :- apja(Y, X), fiu(X), anyja(Y, X), fiu(X).
```

```
lanya(X, Y) :- apja(Y, X), lany(Y).
lanya(X, Y) :- anyja(Y, X), lany(Y).
nagyi(X, Z) :- apja(Y, Z), anyja(X, Y).
nagyi(X, Z) :- anyja(Y, Z), anyja(X, Y).
papi(X, Z) :- apja(Y, Z), apja(X, Y).
papi(X, Z) :- anyja(Y, Z), apja(X, Y).
ose(X, W) :- apja(X, W).
ose(X, W) :- anyja(X, W).
ose(X, W) :- apja(Y, W), ose(X, Y).
ose(X, W) :- anyja(Y, W), ose(X, Y).
goal
 nagyi(Ki, zoltan).
```

****

3. gyakorlat
(konzultáció)

LISTA.PRO

```
domains
  elem = integer
  list = elem*
predicates
  urese(list)
  nondeterm eleme(elem, list)
  illeszthetoe(list, list).
  nondeterm reszhalmazae(list, list)
  pluszegy(integer, integer)
  szumma(list, integer)
clauses
  urese([]).
  urese([_|_]) :- fail.
  eleme(_,[]) :- fail.
  eleme(X,[X|_]).
  eleme(X,[_|Ys]) :- eleme(X,Ys).
  illeszthetoe([],[]).
  illeszthetoe([X|Xs],[Y|Ys]):-X=Y,
                    illeszthetoe(Xs,Ys).
  reszhalmazae([],_).
  reszhalmazae([X|Xs],Ys):-eleme(X,Ys),
                    reszhalmazae(Xs,Ys).
  pluszegy(N,X):-X=N+1.
  szumma([],0).
  szumma([N|Ns],X):-szumma(Ns,Y),X=N+Y.
goal
  pluszegy(N,6).
```

PELDA_CUT.PRO

```
Domains
  nev, mit = symbol
Predicates
  nondeterm szeret(nev, mit)
Clauses
% a ket adam szereti a bort-bol valassz egyet
  szeret(adam,  bor).          % van megoldas, X = janos
%  szeret(adam,  bor):-!.      % nincs megoldas
  szeret(janos, bor).
  szeret(janos, konyv).

Goal
  szeret(X, bor), szeret(X, konyv).
```

PLUSZEGY.PRO

```
domains
  elem = integer
  list = elem* % elem tagokbol allo lista
predicates
  nondeterm eleme(elem, list)
  egyenlo(list,list).
  pluszegy(integer, integer).
  szumma(list, integer).
clauses
  eleme(_,[]) :- fail.
  eleme(X,[X|_]).
  %eleme(X,[_,X|_]).
  eleme(X,[_|Ys]) :- eleme(X,Ys).
  egyenlo([],[]).
  egyenlo([X|Xs],[X|Ys]):-egyenlo(Xs,Ys).
  pluszegy(N,X):-X=N+1.
  szumma([],0). %szumma([],X) :- X=0.
  szumma([N|Ns],X) :- szumma(Ns,X2),X=N+X2.
goal
  szumma([1,2,3],X).
```

PRIMGYURU.PRO

```
domains
  ints = integer*

predicates
  nondeterm primgyuru(ints)
  nondeterm primgyuru2(integer ,integer, ints)
  nondeterm prim(integer)
  nondeterm forcik_nemoszthato(integer, integer, integer)
  integer int_sqrt(integer)
%  int_sqrt(integer, integer)

%facts - beolv
%  nondeterm beolv_primgyuru(ints)

clauses
  primgyuru([X | Xs]):-
        primgyuru2(X, X, Xs).

  primgyuru2(X, Y, [Z]):-
        P1 = Y+Z,
        P2 = X+Z,
        prim(P1),
        prim(P2).
  primgyuru2(X, Y, [Z | Zs]):-
        P = Y+Z,
        prim(P),
        primgyuru2(X, Z, Zs).

  prim(1).
  prim(2).
  prim(N):-
        Sqrt = int_sqrt(N),
%       int_sqrt(N, Sqrt),
        forcik_nemoszthato(N, 2, Sqrt).

  forcik_nemoszthato(N, With, To):-
```

```
        With >= To,
        N mod With <> 0.
 forcik_nemoszthato(N, With, To):-
        N mod With <> 0,
        With2 = With + 1,
        forcik_nemoszthato(N, With2, To).

% a fugveny feje: integer int_sqrt(integer)
 int_sqrt(N, Return):-
        Return=cast(integer,sqrt(N)).

GOAL
  primgyuru([1,4,7,4]).
```

SUMMA.PRO

```
domains
  ints = integer*
 predicates
   summa(ints, integer)
 clauses
   summa([],  N) :- N = 0.
   summa([X | Xs], N) :-
     summa(Xs, M),
     S = X + M,
     S = N.
goal
  summa([1, 2, 3], N)
```

SZERET.PRO

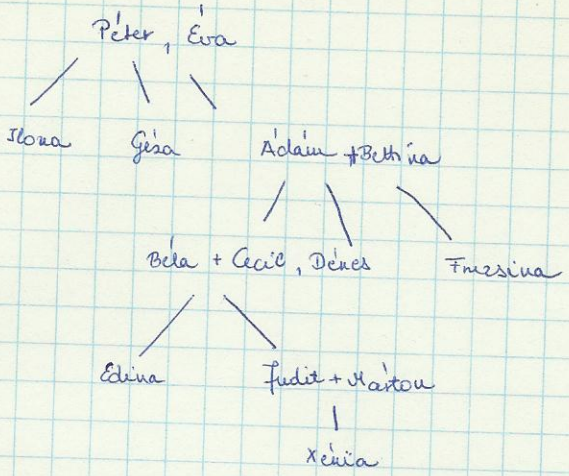
```
domains

predicates
  nondeterm szeret(symbol, symbol)
  olvas(symbol)
clauses
  szeret(jozsi, konyv).
  szeret(jozsi, foci).
  szeret(boske, konyv).
  szeret(A, konyv) :- olvas(A). %szabaly
  % szeret(A, konyv) <= olvas(A)
  olvas(peti).
goal
  szeret(Ki, konyv).
```

SZERETI.PRO

```
domains
  nev = symbol
  mit = symbol
 predicates
   nondeterm szereti(nev, mit)
   olvas(nev)
   focizik(nev)
clauses
  olvas(peter).
  focizik(adam).
  szereti(X, fuci) :- focizik(X).
  szereti(X, konyv) :- olvas(X).
goal
  szereti(peter, M).
```

1. gyakorlat



- nagyapja (X, Y)
- testvére (X, Y)
- bátyja (X, Y)
- őse (X, Y)
- őscsaládja (X, Y)

```
X
| apja
2
| szülő
X
```

testver(X,Y) :- szulo(Z,X), szulo(Z,Y).

```
X
| szülő
2
:  } őse
:
X
```

peter ősc xenia