

Logikai programozás

9.

EGY KIS ISMERET-TÁGÍTÁS

Programok:

- struktura.pl
- operatorok.pl, precedencia.pl
- útvonaltervezés
- send_more_money (smm1.pl, smm2.pl)
- clp_pelda.pl
- barkochba

STRUKTÚRÁK

A Prologban az adatbázis tény-halmazként reprezentálható.
De ezek a tények strukturáltak is lehetnek.

Pl.:

```
% család(apa, anya, gyerekek).
```

```
csalad(szemely(nagy, adam, szul(1970, 03, 17)),  
       szemely(kiss, eva, szul(1973, 07, 22)),  
       [szemely(nagy, peter, szul(1995, 05, 13)),  
        szemely(nagy, judit, szul(1998, 03, 28)),  
        szemely(nagy, adam, szul(2003, 11, 02))] ).
```

STRUKTÚRÁK – NÉHÁNY LEKÉRDEZÉS

1. Kétyerekes családok nevének kiírása:

```
forall((csalad(Apa, Anya, Gyerekek), length(Gyerekek, 2),  
       Apa = szemely(Veznev, _)), writeln(Veznev)).
```

2. Azok a gyerekek, akiknek édesapja 1 évvel idősebb az édesanyjánál:

```
forall((csalad(szemely(_,_), szul(ApaEv, _)),  
       szemely(_,_), szul(AnyaNev, _)), Gyerekek),  
       AnyaNev - ApaEv =:= 1,  
       member(Gyerek, Gyerekek)),  
       writeln(Gyerek)).
```

STRUKTÚRÁK – NÉHÁNY LEKÉRDEZÉS

3. Összes szülő névsora

Ehhez előbb egy szabály a szülő fogalmára:

```
szulo(Vez,Ker, szul(Ev,Ho,Nap)) :-  
    család(szemely(Vez,Ker, szul(Ev,Ho,Nap)),_,_);  
    család(_,szemely(Vez,Ker, szul(Ev,Ho,Nap)),_).
```

```
forall(szulo(Veznev,Kernev, _),  
       writef('%w %w\n',[Veznev, Kernev])).
```

OPERÁTOROK, PRECEDENCIA

Prolog jelölés:

predikatumNev(argumentumok) – prefix jelölés

E szerint a szintaktika alapján a + b -t így kellene írunk:

+ (a,b)

Kialakíthatunk infix jelölést is (ilyen az a + b).

Az eljárás:

op(+Precedence, +Type, :Name)

A precedencia a kiértékelési sorrendet határozza meg, a típus jelzi, hogy hova kerüljön a predikátumnév, a név az operátor nevét jelzi. (A név helyett lista is írható.)

Bővebben ld. help.

OPERÁTOROK, PRECEDENCIA

1. példa:

```
:-op(800, xfx, [nagyszuloje,szuloje]).
```

```
szuloje(adam,pisti).
```

```
szuloje(adam,mari).
```

```
jani szuloje bela.
```

N nagyszuloje Gy :- N szuloje Sz, Sz szuloje Gy.

Lekérdezés pl.: ?- N nagyszuloje Gy.

vagy

```
?- setof((A,Unokak),setof(B, A nagyszuloje B, Unokak),L).
```

OPERÁTOROK, PRECEDENCIA

2. példa:

Játszadozzon el a precedencia.pl fájjal.

```
:-op(300,yfx,(-)).
```

```
%:-op(100,yfx,(-)).
```

```
%:-op(300,xfy,(-)).
```

```
%:-op(100,xfy,(-)).
```

```
%:-op(300,xfx,(-)).
```

Próbahívás pl.:

X is 5 - 3 - 2.

vagy

X is 5 - 3 ** 2.

OPERÁTOROK, PRECEDENCIA

SWI-Prologban a precedencia értékek 0 és 1200 között változhatnak, minél kisebb, annál erősebb.

Lekérdezése pl.:

```
?- current_op(Prec,Assoc,**).  
Prec = 200,  
Assoc = xfx.
```

<http://www.learnprolognow.org/lpnpagel.php?pagetype=html&pageid=lpn-htmlse40>

http://www.cs.ubbcluj.ro/~csatol/log_funk/prolog/slides/4-operators.pdf

http://www.cs.ubbcluj.ro/~csatol/log_funk/prolog/slides/

+ google

ÚTVONALKERESÉS

3. példa:

Próbálja meg továbbfejleszteni az útvonaltervezést.

A `jaratok.pl` modul felhasználásával készítsen olyan útvonaltervező programot, amely segítségével eldönthető, hogy:

- el tudunk-e jutni repülővel egyik városból a másikba,
- adott napon el tudunk-e jutni, stb.

```
jarat(london, edinburgh, [ 9:40 / 10:50 / le2134/ naponta,  
                            11:40 / 12:50 / le2135/ naponta,  
                            18:40/ 19:50 / le2136/  
                            [hef,kedd,sze,csut,pent]]).
```

REJTVÉNYEK

1. példa:

```
  SEND  
+ MORE  
-----  
 MONEY
```

REJTVÉNYEK

smm :-

```
X = [S,E,N,D,M,O,R,Y],  
Digits = [0,1,2,3,4,5,6,7,8,9],  
assign_digits(X, Digits),  
M > 0, S > 0,  
1000*S + 100*E + 10*N + D +  
1000*M + 100*O + 10*R + E =:=  
10000*M + 1000*O + 100*N + 10*E + Y,  
write(X).
```

`assign_digits([], _).`

```
assign_digits([D|Ds], List) :- select(D, List, NewList),  
                               assign_digits(Ds, NewList).
```

REJTVÉNYEK

Ugyanez a constraint logic csomag segítségével:

```
:- use_module(library(clpfd)).
```

```
smm :-
```

```
  X = [S,E,N,D,M,O,R,Y],
```

```
  X ins 0 .. 9,
```

```
  M #> 0, S #> 0,
```

```
    1000*S + 100*E + 10*N + D +
```

```
    1000*M + 100*O + 10*R + E #=
```

```
    10000*M + 1000*O + 100*N + 10*E + Y,
```

```
  all_distinct(X),
```

```
  label(X),
```

```
  write(X).
```

ELEMI PICI PÉLDA

CLP (Constraint logic programming – korlátlogikai progr.)

PI.:

```
haromszog(A,B,C) :- A >= 0, B >= 0, C >= 0,
```

```
                  A + B >= C, A + C >= B, B + C >= A.
```

?- haromszog(3,4,5). – a válasz: true.

?- haromszog(3,4,C). – nem tudja megoldani.

A CLP korlátot szabhat C számára.

Esetünkben $1 \leq C \leq 7$ korlát lehet – ez lesz az eredmény.

(library(clpfd): Constraint Logic Programming over Finite Domains)

REJTVÉNYEK

2. példa:

Három embert mutatunk be, Aladárt, Bélát és Balázst.

Egyikük asztalos, a másik bádogos, a harmadik agronómus. Egyikük Budapesten, a másik Békéscsabán, a harmadik Aszódon lakik, Hogy ki hol lakik, és mi a foglalkozása, azt találja ki Ön.

Annyit azonban elmondhatunk, hogy

- Balázs csak ritkán látogat a fővárosba, pedig minden rokona Budapesten lakik;
- két olyan ember is van a társaságban, akinek foglalkozása és lakhelye ugyanazzal a betűvel kezdődik, mint a neve;
- az asztalos felesége Balázs húga.

REJTVÉNYEK

```
:- use_module(library(clpfd)).
```

```
start :- All = [Aladar,Bela,Balazs,  
               Asztalos,Badogos,Agronomus,  
               Budapest,Bekescsaba,Aszod],
```

```
  All ins 1..3,
```

```
  all_different([Aladar,Bela,Balazs]),
```

```
  all_different([Asztalos,Badogos,Agronomus]),
```

```
  all_different([Budapest,Bekescsaba,Aszod]),
```

```
  label(All),
```

```
  kiir(1,All),kiir(2,All),kiir(3,All),fail.
```

```
start.
```

REJTVÉNYEK

```
kiir(N,[Aladar,Bela,Balazs,
Asztalos,Badogos,Agronomus,
Budapest,Bekescsaba,Aszod]):-
nth1(N1,[Aladar,Bela,Balazs],N),
nth1(N1,['Aladár','Béla','Balázs'],Nev),
nth1(N2,[Budapest,Bekescsaba,Aszod],N),
nth1(N2,['Budapest','Békéscsaba','Aszód'],Hely),
nth1(N3,[Asztalos,Badogos,Agronomus],N),
nth1(N3,['asztalos','bádogos','agronómus'],Munka),
writef('%w\t%w\t%w\n',[Nev,Hely,Munka]).
```

Így nagyon sok hármast írna ki – korlátozni kell.

REJTVÉNYEK

A labell(All) hívás elé még beírjuk az ismert, ill. kikövetkeztetett feltételeket:

```
Balazs #\= Budapest,
Asztalos #\= Balazs,
Asztalos #= Budapest,
```

Mivel Balázs nem budapesti így azt is rögzíthetjük, hogy nem ugyanahhoz a hármashoz tartoznak, így pl.

```
Balazs #=1,
Budapest #=2,
```

Még mindig több megoldás van, de már lényegesen szűkült a számuk.

REJTVÉNYEK

Ha még azt is figyelembe vesszük, hogy két személy neve, foglalkozása és lakóhelye is azonos betűvel kezdődik, akkor még ezeket a feltételeket is hozzávehetjük:

```
Balazs #= Badogos,
Balazs #= Bekescsaba,
Aladar #= Aszod,
Aladar #= Agronomus,
```

Így ugyan „kézzel” megoldottuk a feladatot, a Prolog csak annyit tesz, hogy ellenőrzi a megoldást, de

a/ az ellenőrzés is fontos

b/ van, amikor a megoldás megtalálásában is segít.

BARKOCHBA

ld. BarKochba.pl, BarKochba2.pl