

# Nagyhatékonyságú deklaratív programozás

Vizsga-minta, 2012. január.

1. Tekintsd az alábbi FD-predikátumot:

```
pred(X, Y) +:  
  Y in (inf .. X) \ / (inf .. -X).
```

a. Határozd meg `pred/2` jelentését, azaz az általa kijelölt relációt (például úgy, hogy aritmetikai és logikai korlátokkal felírsz egy a fenti FD-predikátummal azonos jelentésű relációt).

b. Írd fel a fenti FD-klóz **mindkét irányban** szűkítő, **monoton** változatát.

c. Írd fel a `pred/2` FD-predikátum `+`? nyakjelű **antimonoton** klózát. (20 pont)

2. Nevezzünk egy számlistát nullsornak, ha egy pozitív számmal kezdődik, és ezt pontosan annyi nulla követi, amennyi a szám értéke.

Írd meg a következő relációt a `library(clpfd)` segítségével! Ne használj választási pontokat (spekulatív diszjunkciót)! Nem kell tartomány-szűkítésre törekedned.

`nullsoros(L)`: Az `L` lista nem más mint valahány nullsor egymás után fűzése (`L` korlát-változók listája). (20 pont)

Példák:

```
| ?- length(L,3), nullsoros(L).  
  L = [2,0,0] ? ; no
```

```
| ?- length(L,6), nullsoros(L), reverse([0|L],[_R]), nullsoros(R).  
  L = [1,0,1,0,1,0], R = [1,0,1,0,1,0] ? ; no
```

```
| ?- L = [_A,_,_,_A,_,_], nullsoros(L).  
  L = [2,0,0,2,0,0] ? ; no
```

```
| ?- L = [_,_,_C,_,_,_], nullsoros(L), _C #> 0.  
  L = [1,0,_C,0,_A,0], _C in 1..3, _A in 0..1 ? ; no
```

3. Globális korlátként írd meg egy `egyseg(L)` predikátumot, amely azt fejezi ki, hogy az L lista elemei között **pontosan** egy negatív szám van. L korlát-változók listája. Figyelj arra, hogy a globális korlát kilépjen, mielőtt a tárból következik a feltétel igaz vagy hamis volta (mint az alábbi példák mindegyikében)!

(25 pont)

Példák:

```
| ?- egyseg([A,B,C]), C #>= 2, B #> 0.
      A in inf.. -1, B in 1..sup, C in 2..sup ?
```

```
| ?- egyseg([A,B,C]), C #=< -2.
      A in 0..sup, B = 0..sup, C in inf.. -2 ?
```

```
| ?- egyseg([A,B,C]), C #< 0, B = -2.
      no
```

```
| ?- egyseg([A,B,C]), C #> 0, B = 2, A #> C.
      no
```

4. Tekintsd az alábbi Mercury programot! Add meg a `q/4` predikátum-deklarációját és az összes lehetséges predikátummód-deklarációját (azaz írd fel az argumentumok módjait minden a `p/3` módjai által megengedett kombinációban és állapítsd meg, milyen determinizmus tartozik hozzá)! Vedd figyelembe, hogy a fordító:

- a „túlságosan behelyettesített” argumentumokat egy új változó és egy egyenlőségvizsgálat bevezetésével ki tudja küszöbölni,
- a klóztörzsben a hívások sorrendjét át tudja rendezni.

(10 pont)

A felhasznált `p/3` predikátumra vonatkozó deklarációk:

```
:- pred p(list(T), list(T), list(T)).
:- mode p(in, in, in) is semidet.
:- mode p(in, in, out) is det.
:- mode p(in, out, in) is semidet.
:- mode p(out, out, in) is multi.
```

A program:

```
q(A, B, C, D) :-
    p(A, E, D),
    p(B, C, E).
```

5. Tekintsd az alábbi CHR programot és a hozzá tartozó alábbi célsorozatot!

```
:- use_module(library(chr)).
handler min.
:- op(700, xfx, leq).
constraints (leq)/2, min/3.

min1 @ min(X, X, Y) <=> X = Y.
min2 @ min(X, Y, Z) ==> Z leq X, Z leq Y.
lm1  @ Y leq X \ min(X, Y, Z) <=> Y=Z.
lm2  @ X leq Y \ min(X, Y, Z) <=> X=Z.
ref1 @ X leq X <=> true.
ants @ X leq Y, Y leq X <=> X = Y.
idem @ X leq Y \ X leq Y <=> true.
tran @ X leq Y, Y leq Z ==> X leq Z.

| ?- min(A, B, C), min(A, C, B).
```

Az alábbi táblázatba írd be a fenti cél futását, lépésenként megadva, hogy milyen címkéjű szabály tüzel és hogyan alakul a tár! Kövesd a példát és a jelmagyarázatot! (10 pont)

Jelmagyarázat:  új, tüzelő, ~~törölt~~ korlát

tüzelő szabály	tárban levő korlátok
<i>célsorozatból</i>	<input type="text"/> $\min(A,B,C)$
min2	<u><math>\min(A,B,C)</math></u> <input type="text"/> $C \leq A$ <input type="text"/> $C \leq B$



# Nagyhatékonyságú deklaratív programozás

Vizsga-minta javítókulcsa, 2012. január

1. Tekintsd az alábbi FD-predikátumot:

```
pred(X, Y) +:  
  Y in (inf .. X) \\/ (inf .. -X).
```

a. Határozd meg `pred/2` jelentését, azaz az általa kijelölt relációt (például úgy, hogy aritmetikai és logikai korlátokkal felírsz egy a fenti FD-predikátummal azonos jelentésű relációt).

b. Írd fel a fenti FD-klóz **mindkét irányban** szűkítő, **monoton** változatát.

c. Írd fel a `pred/2` FD-predikátum `-?` nyakjelű **antimonoton** klózát.

(20 pont)

## Megoldás

a. `pred(X, Y) :- Y #=< X #\/ Y #=< -X.`

vagy

```
pred(X, Y) :- Y #=< max(X, -X).
```

vagy

```
pred(X, Y) :- Y #=< abs(X).
```

(3 pont)

b. `pred(X, Y) +:`

```
  Y in (inf .. max(X)) \\/ (inf .. (-min(X))),
```

```
  X in \ (((-min(Y))+1) .. (min(Y)-1)).
```

Az első indexikálisra:

(3 pont)

A második indexikálisra:

(7 pont)

c. `pred(X, Y) -?`

```
  Y in ((max(X)+1) .. sup) /\ (((-min(X))+1) .. sup).
```

(7 pont)

2. Nevezzünk egy számlistát nullsornak, ha egy pozitív számmal kezdődik, és ezt pontosan annyi nulla követi, amennyi a szám értéke.

Írd meg a következő relációt a `library(clpfd)` segítségével! Ne használj választási pontokat (spekulatív diszjunkciót)! Nem kell tartomány-szűkítésre törekedned.

`nullsoros(L)`: Az `L` lista nem más mint valahány nullsor egymás után fűzése (`L` korlát-változók listája).

(20 pont)

Példák:

```
| ?- length(L,3), nullsoros(L).
```

```
  L = [2,0,0] ? ; no
```

```
| ?- length(L,6), nullsoros(L), reverse([0|L],[_R]), nullsoros(R).
```

```
  L = [1,0,1,0,1,0], R = [1,0,1,0,1,0] ? ; no
```

```
| ?- L = [_A,_,_,_A,_,_], nullsoros(L).
```

```
  L = [2,0,0,2,0,0] ? ; no
```

```
| ?- L = [_,_,_C,_,_,_], nullsoros(L), _C #> 0.
```

```
  L = [1,0,_C,0,_A,0], _C in 1..3, _A in 0..1 ? ; no
```

## Megoldás

```

%% nullsoros(L): Az L lista nullsorok sorozata
nullsoros([]).
nullsoros([H|T]) :-
    H #> 0,
    nullsoros(T, H).

%% nullsoros(L, N): Az L lista elején N db 0 áll,
%% amit nullsorok sorozata követ. (N >= 0)
nullsoros([], 0).
nullsoros([H|T], N) :-
    H #>= 0,
    N #= 0 #<=> H #> 0,
    N1 #= max(H, N-1),
    nullsoros(T, N1).

```

3. Globális korlátként írj meg egy **egyneg(L)** predikátumot, amely azt fejezi ki, hogy az L lista elemei között **pontosan** egy negatív szám van. L korlát-változók listája. Figyelj arra, hogy a globális korlát kilépjen, mielőtt a tárból következik a feltétel igaz vagy hamis volta (mint az alábbi példák mindegyikében)!

(25 pont)

### Megoldás

```

egyneg(L) :-
    minmax_susps(L, S),
    fd_global(egyneg(L), L, S).

:- multifile clpfd:dispatch_global/4.
clpfd:dispatch_global(egyneg(_), L0, L, Actions) :-
    egyneg_megoldo(L0, L, Actions).

egyneg_megoldo(L0, L, Actions) :-
    szuro(L0, L, 0, Neg),
    ( Neg = 1 -> Actions = [exit|Acts0], allnonneg(L, Acts0)
    ; Neg = 0 ->
      ( L = [] -> Actions = [fail]
      ; L = [X] -> Actions = [exit,X in inf.. -1]
      ; Actions = []
      )
    ; Actions = [fail]
    ).

% szuro(L0, L, NO, N): L az L0-ból az n db biztosan negatív és (akárhány db)
% biztosan nem negatív FD változó elhagyásával jön létre, N = NO+n.
szuro([], [], NO, NO).
szuro([H|T], L, NO, N) :-
    fd_min(H, Min), fd_max(H, Max),
    ( number(Min), Min >= 0 -> N1 = NO, L = L1
    ; number(Max), Max < 0 -> N1 is NO+1, L = L1
    ; L = [H|L1], N1 = NO
    ),
    szuro(T, L1, N1, N).

allnonneg([], []).
allnonneg([H|T], [H in 0..sup|T0]) :-
    allnonneg(T, T0).

```

```

minmax_susps([], []).
minmax_susps([H|T], [minmax(H)|VT]) :-
    minmax_susps(T, VT).

```

4. Tekintsd az alábbi Mercury programot! Add meg a q/4 predikátum-deklarációját és az összes lehetséges predikátummód-deklarációját (azaz írd fel az argumentumok módjait minden a p/3 módjai által megengedett kombinációban és állapítsd meg, milyen determinizmus tartozik hozzá)! Vedd figyelembe, hogy a fordító:

- a „túlságosan behelyettesített” argumentumokat egy új változó és egy egyenlőségvizsgálat bevezetésével ki tudja küszöbölni,
- a klóztörzsben a hívások sorrendjét át tudja rendezni.

(10 pont)

A felhasznált p/3 predikátumra vonatkozó deklarációk:

```

:- pred p(list(T), list(T), list(T)).
:- mode p(in, in, in) is semidet.
:- mode p(in, in, out) is det.
:- mode p(in, out, in) is semidet.
:- mode p(out, out, in) is multi.

```

A program:

```

q(A, B, C, D) :-
    p(A, E, D),
    p(B, C, E).

```

## Megoldás

```

:- pred q(list(T), list(T), list(T), list(T)).

:- mode q(in, in, in, in) is semidet.
:- mode q(in, in, in, out) is det.
:- mode q(in, in, out, in) is semidet.
% mode q(in, in, out, out) nem megengedett!
:- mode q(in, out, in, in) is nondet.
% mode q(in, out, in, out) nem megengedett!
:- mode q(in, out, out, in) is nondet.
% mode q(in, out, out, out) nem megengedett!
:- mode q(out, in, in, in) is nondet.
% mode q(out, in, in, out) nem megengedett!
:- mode q(out, in, out, in) is nondet.
% mode q(out, in, out, out) nem megengedett!
:- mode q(out, out, in, in) is nondet.
% mode q(out, out, in, out) nem megengedett!
:- mode q(out, out, out, in) is multi
% mode q(out, out, out, out) nem megengedett!

```

A helyes :- pred deklarációra 1 pont, minden helyes :- mode deklarációra 1 pont.

5. Tekintsd az alábbi CHR programot és a hozzá tartozó alábbi célsorozatot!

```

:- use_module(library(chr)).

:- op(700, xfx, leq).

:- chr_constraint (leq)/2, min/3.

min1 @ min(X, X, Y) <=> X = Y.
min2 @ min(X, Y, Z) ==> Z leq X, Z leq Y.
lm1 @ Y leq X \ min(X, Y, Z) <=> Y=Z.
lm2 @ X leq Y \ min(X, Y, Z) <=> X=Z.
refl @ X leq X <=> true.
ants @ X leq Y, Y leq X <=> X = Y.
idem @ X leq Y \ X leq Y <=> true.
tran @ X leq Y, Y leq Z ==> X leq Z.

| ?- min(A, B, C), min(A, C, B).

```

Az alábbi táblázatba írd be a fenti cél futását, lépésenként megadva, hogy milyen címkéjű szabály tüzel és hogyan alakul a tár! Kövesd a példát és a jelmagyarázatot!

(10 pont)

Jelmagyarázat: új, tüzelő, ~~törölt~~ korbát

**Megoldás**

tüzelő szabály	tárban levő korlátok
<i>célsorozatból</i>	<span style="border: 1px solid black; padding: 2px;">min(A,B,C)</span>
min2	<u>min(A,B,C)</u> <span style="border: 1px solid black; padding: 2px;">C leq A</span> <span style="border: 1px solid black; padding: 2px;">C leq B</span>
<i>célsorozatból</i>	min(A,B,C) C leq A C leq B <span style="border: 1px solid black; padding: 2px;">min(A,C,B)</span>
min2	min(A,B,C) C leq A C leq B <u>min(A,C,B)</u> <span style="border: 1px solid black; padding: 2px;">B leq A</span> <span style="border: 1px solid black; padding: 2px;">B leq C</span>
ants	min(A,B,C) C leq A <del>C leq B</del> min(A,C,B) B leq A <del>B leq C</del> <span style="border: 1px solid black; padding: 2px;">C = B</span>
idem	min(A,B,B) <u>B leq A</u> min(A,B,B) <del>B leq A</del> C = B
lm1	<del>min(A,B,B)</del> <u>B leq A</u> min(A,B,B) C = B
lm1	<u>B leq A</u> <del>min(A,B,B)</del> C = B
	B leq A C = B

Rontott lépésenként 1-2 pont levonás.