Nagyhatékonyságú logikai programozás — feladatgyűjtemény

Szeredi Péter, Benkő Tamás

1. A 2000. évi őszi szemeszter feladatai

1.1. Vizsgafeladatok

1. Tekintsd a következő FD-predikátumot:

```
pred(X, Y) +:
        Y in (inf..200) /\ (inf..X*10).
```

a. Határozd meg `pred/2` jelentését, azaz az általa kijelölt relációt.

b. Írd fel a fenti FD-klóz (mindkét irányban) tartomány-szűkítő változatát.

c. Írd fel a `pred/2` FD-predikátum `+?` nyakjelű, tartomány-levezethetőséget biztosító klózát.

2. Globális korlátként írj meg egy `legf1poz(L)` predikátumot, amely azt fejezi ki, hogy az L lista elemei között **legfeljebb** egy pozitív (> 0) szám van.

Példák:

```
| ?- legf1poz([A,B,C]), C #>=2.
        A in inf..0, B in inf..0, C in 2..sup ?

| ?- legf1poz([A,B,C]), C #=< 0, B = 0.
        A in inf..sup, B = 0, C in inf..0 ?

| ?- legf1poz([A,B,C]), C #>= 1, B = 2.
        no
```

3. Nevezzünk egy L listát N-változatosnak, ha L-nek nincs csupa azonos elemből álló N hosszúságú folytonos részlistája, azaz bármely N szomszédos elem között legalább két különböző érték fordul elő.

Írd meg a következő relációt a `library(clpfd)` segítségével! Ne használj választási pontokat (spekulatív diszjunkciót)!

`valto(L, N)`: Az `L` lista `N`-változatos.

Példák:

```
| ?- L = [_C,1,_,2,_C], domain(L, 1, 3), valto(L, 2).
        L = [3,1,3,2,3] ?

| ?- L = [_,1,_,_,2,_], domain(L, 1, 2), valto(L, 2).
        L = [2,1,2,1,2,1] ?

| ?- L = [_,1,_,2,_], domain(L, 1, 2), valto(L, 2).
        no
```

4. Tekintsd az alábbi Mercury programot! Add meg a felhasznált típusok deklarációját, a hiányzó predikátum-deklarációkat, és azokat a predikátummód-deklarációkat, amelyek a `p1` és `p2` predikátumokhoz szükségesek! Add meg a determinizmusokat is! Ne hagyatkozz implikált módokra!

A program:

```
% zip(P, A, B, C): Az A és a B egyforma hosszú listák azonos indexű
% elemeit a P predikátum szerint komponálva es ezeket listába gyűjtve
% kapjuk a C listát.
:- pred zip(pred(T, T, T), list(T), list(T), list(T)).
zip(_, [], [], []).
zip(P, [H1|T1], [H2|T2], [H3|T3]) :-
        call(P, H1, H2, H3),
        zip(P, T1, T2, T3).

p1(A, B, C) :-
      zip((pred(X::in,Y::out,Z::in) is semidet :- X-Y = Z), A, B, C).

p2(A, B, C) :-
      zip((pred(X::in,Y::in,Z::out) is det :- [X|Y] = Z), A, B, C).
```

5. Tekintsd az alábbi CHR programot.

```
:- use_module(library(chr)).
handler bool.
constraints or/3, neg/2.

or1@    or(0,X,Y) <=> Y=X.
or2@    or(X,0,Y) <=> Y=X.
or3@    or(X,Y,0) <=> X=0,Y=0.
or4@    or(1,X,Y) <=> Y=1.
or5@    or(X,1,Y) <=> Y=1.
or6@    or(X,X,Z) <=> X=Z.

neg1@   neg(0,X) <=> X=1.
neg2@   neg(X,0) <=> X=1.
neg3@   neg(1,X) <=> X=0.
neg4@   neg(X,1) <=> X=0.
neg5@   neg(X,X) <=> fail.
neg6@   neg(X,Y) \ or(X,Y,Z) <=> Z=1.
neg7@   neg(Y,X) \ or(X,Y,Z) <=> Z=1.
```

Lépésenként írd le az alábbi cél futását, megadva melyik szabály tüzel és hogyan alakul a tár.

```
| ?- or(A, B, C), neg(A, B), neg(C, D), or(A, C, B).
```

2. A 2000. évi tavaszi szemeszter feladatai

2.1. Vizsgafeladatok

1. Határozd meg a változók tartományának változását az alábbi célsorozat végrehajtása során! Lépésenként írd le, hogy az `a`, `b` és `c` betűkkel jelzett korlátokból milyen démonok keletkeznek, ezek mikor ébrednek fel milyen szűkítést végezve, és mikor fejezik be működésüket!

```
nevtelen(X, Y, Z) +:
      Z in ((min(X)..sup) /\ (min(Y)..sup)) /\
           ((inf..max(X)) \/ (inf..max(Y))).
```

```
| ?- domain([X,Y,Z], 2, 5), /*a:*/ X+Y#=6,
                            /*b:*/ nevtelen(X,Y,Z),
                            /*c:*/ Y#>3.
```

2. Írj egy `optimum/7` eljárást a `library(clpq)` segítségével, amellyel egy kéterőforrásos lineáris optimalizálási feladatot lehet megoldani!

$$\text{optimum}(m,a,x,b,y,z,p) \Leftrightarrow \sum_i m_i * x_i \leq a \wedge \sum_i m_i * y_i \leq b \wedge p = \max_m \sum_i m_i * z_i$$

`optimum/7` első argumentuma egy lista, amely az egyes termékek mennyiséget tartalmazza, második argumentuma az első erőforrás kapacitása, a harmadik egy lista, melynek *i*. eleme az *i*. termék által igényelt kapacitás. A negyedik és az ötödik argumentum olyan mint az előző kettő, csak a második erőforrásra vonatkozik. A hatodik argumentum *i*. eleme az *i*. termék profitja, a hetedik a maximális összprofit. Az első és a hetedik argumentum kimenő, a többi bemenő.

Példa:

```
| ?- optimum([X,Y,Z], 7, [1,5,2], 16, [3,8,5], [5/2,7,9/3], P).
                  P = 95/7, X = 24/7, Y = 5/7, Z = 0 ? ;  no
```

3. Definiáld az `eq1(L)` globális korlátot, amely 0-1 értékű változók listájára biztosítja, hogy a lista elemei között pontosan egy darab 1-es van. Csak a hiányzó részt kell megírnod.

```
% L egy bitvektor, amiben pontosan 1 db 1-es van.
eq1(L) :-
        domain(L, 0, 1),
        val_susps(L, S),
        fd_global(eq1, L, S).

val_susps([], []).
val_susps([H|T], [val(H)|VT]) :-
        val_susps(T, VT).
```

Példák:

```
| ?- eq1([A,B,C]), C = 1.
                        A = 0, B = 0, C = 1 ? ;  no
| ?- eq1([A,B,C]), C = 0.
                        C = 0, A in 0..1, B in 0..1 ? ;  no
| ?- eq1([A,B,C]), C = 0, A = 0.
                        A = 0, B = 1, C = 0 ? ;  no
| ?- eq1([A,B,C]), C = 1, A = 1.
                        no
```

4. A feladat a 8. rejtvenyfejtő világbajnokság egyik rejtvényének alaplépését fogalmazza meg. A rejtvény a következő:

Egy négyzetrács minden sorába és minden oszlopába az `[1,`*n*`]` számok egy permutációját kell elhelyezni, ahol n a rács szélessége. A számok felhőkarcolók magasságát reprezentálják, a nagyobb számok magasabbakat. A széleken levő számok adottak, és azt mutatják, hogy **hány felhőkarcoló látható** az adott irányból.

Alább egy rejtvény és a megoldása (a helyesen kitöltött pálya) látható.

	3	1	2	2	
2					3
2					1
1					3
3					2
	2	3	1	2	

	3	1	2	2	
2	2	4	3	1	3
2	3	1	2	4	1
1	4	3	1	2	3
3	1	2	4	3	2
	2	3	1	2	

Írd meg a fenti rejtvényhez kapcsolódó relációt a `library(clpfd)` segítségével! Ne használj választási pontokat (spekulatív diszjunkciót)!

`emelkedo(`*`Valtozok`*`, `*`Hossz`*`, `*`Lathatok`*`)`: *`Valtozok`* az 1 és *`Hossz`* közötti egész számok permutációját tartalmazó lista, amelyben a látható elemek száma *`Lathatok`* (láthatónak hívunk egy elemet, ha a listában előtte álló elemek mindegyikénél nagyobb).

Példák:

```
| ?- emelkedo(L,3,3), labeling([], L).
                        L = [1,2,3] ? ;  no
| ?- emelkedo(L,3,2), labeling([], L).
                        L = [1,3,2] ? ;  L = [2,1,3] ; L = [2,3,1] ? ;  no
```

5. Tekintsd az alábbi Mercury programot! Add meg a felhasznált típusok deklarációját és azokat a predikátummód-deklarációkat, amelyek a `p1` és `p2` predikátumokhoz szükségesek! Add meg a determinizmusokat is! Ne hagyatkozz implikált módokra!

A program:

```
% transform(P, A, B): Az A fanak a P predikatum szerinti
% transzformaltja a B fa.
:- pred transform(pred(T, T), bfa(T), bfa(T)).
transform(P, level(A), level(B)) :-
        call(P, A, B).
transform(P, ag(A, BFa1, JFa1), ag(B, BFa2, JFa2)) :-
        call(P, A, B),
        transform(P, BFa1, BFa2), transform(P, JFa1, JFa2).
```

A hívási környezet:

```
:- pred p1(bfa(lista)::out, bfa(lista)::out).
p1(A, B) :-
        transform((pred(X::out,Y::out) is multi :- append(X,Y,[1,2,3])), A, B).

:- pred p2(bfa(list(T))::in, bfa(list(T))::out).
p2(A, B) :- transform((pred(X::in,Y::out) is semidet :- [_|Y] = X), A, B).
```

2.2. Gyakorló feladatsor

Az alábbi 1.-4. feladatok megegyeznek az 1999. évi vizsgafeladatokkal.

1. Határozd meg a változók tartományának változását az alábbi célsorozat végrehajtása során! Lépésenként írd le, hogy az `a`, `b` és `c` betűkkel jelzett korlátokból milyen démonok keletkeznek, ezek mikor ébrednek fel milyen szűkítést végezve, és mikor fejezik be működésüket! (Az `X*C#=Y` korlát intervallum-konzisztens.)

```
| ?- domain([X,Y], 1, 5), /*a:*/ Z#>2,
                          /*b:*/ X#=<Y #=> Z#=<Y,
                          /*c:*/ X*2#=Y.
```

2. Írj egy `megold/3` eljárást a `library(clpr)` segítségével, amellyel egy n-ismeretlenes lineáris egyenletet lehet megoldani! `megold/3` első argumentuma egy n*n-es valós mátrix, az egyenletek együtthatóiból álló vektorok listájaként ábrázolva, a második az egyenletek jobb-oldalát megadó n-es valós vektor, a harmadik, kimenő argumentum pedig a szintén n-es megoldásvektor.

Példa:

```
| ?- megold([[1,2],[3,4]], [7,15], V).
                                      V = [1.0,3.0] ?
```

3. Definiáld az `'x>0\/y>0'(X, Y)` korlátot, amely nevének megfelelő jelentésű, és lehetőleg minél erősebb szűkítést biztosít!

a) Definiáld a korlátot FD-predikátummal! Csak a `+:` nyakjelű klózt kell megírnod!

b) Definiáld a relációt globális korlátként!

Példák:

```
| ?- domain([X,Y], -10, 10), 'x>0\/y>0'(X, Y), X#>3.
                                X in 4..10, Y in-10..10 ?

| ?- domain([X,Y], -10, 10), 'x>0\/y>0'(X, Y), X#<1.
                                X in-10..0, Y in 1..10 ?

| ?- domain([X,Y], -10, 10), 'x>0\/y>0'(X, Y), Y#<1.
                                X in 1..10, Y in-10..0 ?
```

4. Írd meg a következő relációt a `library(clpfd)` segítségével! Ne használj választási pontokat (spekulatív diszjunkciót)!

`szakaszok(`*`BitV`*`, `*`N`*`, `*`Hosszak`*`)`: N adott egész, *Hosszak* adott egész-lista, *BitV* 0..1 értékű korlát-változók N hosszú listája. A *BitV* bitvektorban a folytonos 1-esek szakaszainak hosszai a *Hosszak* listát adják. Például `szakaszok([0,1,1,0,1,1,1], 7, [2,3])` igaz.

Példák:

```
| ?- szakaszok(BV, 7, [2,3]).
       BV = [_A,1,_B,_C,1,1,_D],
       _A in 0..1, _B in 0..1, _C in 0..1, _D in 0..1 ?

| ?- szakaszok(BV, 7, [2,3]), BV=[0|_].
       BV = [0,1,1,0,1,1,1] ?
```

5. Tekintsd az alábbi Mercury programot! Add meg a felhasznált típusok deklarációját, a predikátum-deklarációkat, és azokat a predikátummód-deklarációkat, amelyek a `p1`, `p2` és `p3` predikátumokhoz szükségesek! Add meg a determinizmusokat is! Ne hagyatkozz implikált módokra!

A program:

```
ut(Graf,    _,    Hova,   Hova, [Hova], 0) :-
        ele(Graf, Hova, _, _).
ut(Graf, Suly, Honnan, Hova, [Honnan|Ut], Koltseg+S) :-
        ele(Graf, Honnan, Kozben, MGraf),
        S = apply(Suly, el(Honnan, Kozben)),
        ut(MGraf, Suly, Kozben, Hova, Ut, Koltseg).

ele([el(A,B)|G], A, B, G).
ele([E|G], A, B, [E|G1]) :-
        ele(G, A, B, G1).
```

A hívási környezet:

```
:- pred p1(int, int, list(int), int).
:- mode p1(out, out, out, out).
p1(Honnan, Hova, Ut, Koltseg) :-
        ut([el(1,2),el(1,4),el(2,3)], suly, Honnan, Hova, Ut, Koltseg).

:- pred p2(int, int, int).
:- mode p2(out, out, out).
p2(Honnan, Hova, Koltseg) :-
        ut([el(1,2),el(1,4),el(2,3)], suly, Honnan, Hova, [1,2,3], Koltseg).

:- pred p3(int, list(int)).
:- mode p3(in, in).
p3(Hova, Ut) :-
        ut([el(1,2),el(1,4),el(2,3)], suly, 2, Hova, Ut, 4).

:- func suly(el) = int.
suly(el(A,B)) = S :-
        (   A = 1 -> S = 1
        ;   S = A-B
        ).
```

2.3. A gyakorló feladatsor megoldásai

1. feladat

```
domain: X in 1..5, Y in 1..5
     a: Z#>2
        végleg lefut (nem keletkezik démon)
        tár: X in 1..5, Y in 1..5, Z in 3..sup
     b: X#=<Y #<=> B1, Z#=<Y #<=> B2, B1 #=< B2 (kifejtés)
        az első két korlátból fejenként 3 démon keletkezik, az utolsóból 1,
        tár: X in 1..5, Y in 1..5, Z in 3..sup, B1 in 0..1, B2 in 0..1
     c: X*2#=Y
        1 démon keletkezik, szűkíti X és Y tartományát.
        tár: X in 1..2, Y in 2..4, Z in 3..sup, B1 in 0..1, B2 in 0..1
     b: X#=<Y levezethető,
        felébred a ... #<=> B1 pozitív levezethetőségi démon,
        behelyettesíti B1-t
        tár: X in 1..2, Y in 2..4, Z in 3..sup, B1 = 1, B2 in 0..1
     b: B1 #=< B2 felébred, behelyettesíti B2-t
        tár: X in 1..2, Y in 2..4, Z in 3..sup, B1 = 1, B2 = 1
     b: felébred a ... #<=> B2 postázó démon,
        felveszi a Z#=<Y korlátot (nevezzük d-nek)
     d: szűkíti Y-t és Z-t
        tár: X in 1..2, Y in 3..4, Z in 3..4, B1 = 1, B2 = 1
     c: X*2#=Y felébred, behelyettesíti mindkét változóját,
```

```
        befejezi működését
        tár: X = 2, Y = 4, Z in 3..4, B1 = 1, B2 = 1
     d: felébred, nem szűkít, elvben befejezheti működését
        tár: X = 2, Y = 4, Z in 3..4, B1 = 1, B2 = 1
```

2. feladat

```
% skalarszorzat(Ak, Bk, Sz): Ak es Bk vektorok skalarszorzata Sz.
skalarszorzat([], [], 0).
skalarszorzat([C|Ck], [X|Xk], Sz+C*X) :-
        skalarszorzat(Ck, Xk, Sz).

% megold(M, A, X): M X = A, ahol M egy nxn-es matrix,
% A es X n hosszusagu vektorok.
megold([], [], _).
megold([S|Sk], [E|Ek], V) :-
        skalarszorzat(S, V, SV),
        {SV = E},
        megold(Sk, Ek, V).
```

3. feladat

```
'x>0\/y>0'(X, Y) +:
        X in (1..max(Y)) ? (inf..sup) \/ (1..sup),
        Y in (1..max(X)) ? (inf..sup) \/ (1..sup).
```

4. feladat

```
% szakaszok(BitV, N, Hosszak): N adott egész, Hosszak adott
% egész-lista, BitV 0..1 értékű korlát-változók N hosszú listája. A BitV
% bitvektorban a folytonos 1-esek szakaszainak hosszai a Hosszak listát
% adják. Például szakaszok([0,1,1,0,1,1,1], 7, [2,3]) igaz.
szakaszok(BoolV, Szelesseg, HosszL) :-
        length(BoolV, Szelesseg),
        domain(BoolV, 0, 1),
        szakasz_valtozok(HosszL, SzakaszL, 0, Szelesseg),
        pozicio_korlatok(BoolV, 1, SzakaszL).

% szakasz_valtozok(Hk, KHk, K0, Sz): Hk egy diszjunkt szakaszok hosszainak
% listaja. KHk egy K-H alaku parokbol allo szakaszleiro lista, ahol H a Hk
% lista megfelelo eleme, K pedig az adott H hosszu szakasz kezdopozicioja
% (korlat valtozo), felteve, hogy ezt a Hk szakaszlistat a [K0,Sz]
% intervallumba kell elhelyezni.
szakasz_valtozok([], [], _, _).
szakasz_valtozok([H|Hk], [K-H|Kk], K0, Sz) :-
        Felso is Sz-H+1,
        K in 1..Felso, K #> K0,
        K1 #= K+H,
        szakasz_valtozok(Hk, Kk, K1, Sz).

% pozicio_korlatok(Bk, I, SzakaszL): A SzakaszL szakaszleiro lista megfelel
% a Bk bitlistanak, felteve, hogy Bk elso elemenek indexe I.
pozicio_korlatok([], _, _).
pozicio_korlatok([B|Bk], I, SzakaszL) :-
        elofordulasai(SzakaszL, I, BL),
        sum(BL, #=, B),
        I1 is I+1,
        pozicio_korlatok(Bk, I1, SzakaszL).
```

```
% elofordulasai(SzakaszL, I, BL): BL[k]=1, ha a SzakaszL[k] tartalmazza az
% I indexu pontot, 0 egyebkent.
elofordulasai([], _, []).
elofordulasai([K-H|KHk], I, [B|Bk]) :-
        I0 is I-H+1,
        K in I0..I #<=> B,
        elofordulasai(KHk, I, Bk).
```

5. feladat

```
:- type pont == int.
:- type el ---> el(pont,pont).
:- type graf == list(el).
:- type ut   == list(int).

:- pred ut(graf, func(el)=int, pont, pont, ut, int).
:- mode ut(in, func(in)=out is det, out, out, out, out) is nondet. % p1
:- mode ut(in, func(in)=out is det, out, out, in, out) is nondet. % p2
:- mode ut(in, func(in)=out is det, in, in, in, in) is semidet. % p3

:- pred ele(graf, pont, pont, graf).
:- mode ele(in, out, out, out) is nondet.
:- mode ele(in, in, out, out) is nondet.
```

3. Korábbi feladatok

3.1. Az 1998. évi vizsgafeladatok

1. Határozd meg a változók tartományának változását az alábbi célsorozat végrehajtása során! Lépésenként írd le, hogy az egyes korlátokból milyen démonok keletkeznek, ezek mikor ébrednek fel milyen szűkítést végezve, és mikor fejezik be működésüket!

```
?- X in 1..8, Y in 21..40,  Y #>= 20+2*X, element(X, [21,23,26,27,32], Y).
```

2. Írj egy `library(clpfd)` programot az alábbi feladat megoldására!

`pyth(N, X, Y, Z)`: Az `X`, `Y` és `Z` `N` jegyű egész számok Pythagorasi számhármast alkotnak, azaz az első két szám négyzetösszege a harmadik. `N` bemenő paraméter. Az $A = 10^N$ érték kiszámolására SICStus-ban használható az `A is integer(exp(10, N))` eljáráshívás.

Példa:

```
| ?- pyth(1, X, Y, Z).
                                X = 3, Y = 4, Z = 5 ? ;
                                X = 4, Y = 3, Z = 5 ? ;
                                no
| ?-
```

3. Definiáld FD-predikátummal (indexikálisokkal) az `'x=<y=<2x'(X, Y)` constraintet, amely nevének megfelelő jelentésű, és lehetőleg minél erősebb szűkítést biztosít! Írd meg az FD-predikátum mind a négy klózát!

Példák:

```
| ?- X in 3..4, 'x=<y=<2x'(X, Y).
                                X in 3..4, Y in 3..8 ?
| ?- Y in 11..20, 'x=<y=<2x'(X, Y).
```

```
                                  Y in 11..20, X in 6..20 ?
| ?- X in 3..4, #\ 'x=<y=<2x'(X,Y).
                                  X in 3..4, Y in(inf..3)\/(7..sup) ?
| ?- X in 3..4,  Y in (inf..2)\/(9..sup), 'x=<y=<2x'(X,Y) #<=> B.
                                  B = 0, X in 3..4, Y in(inf..2)\/(9..sup) ?
| ?- X in 3..4,  Y in 4..6, 'x=<y=<2x'(X,Y) #<=> B.
                                  B = 1, X in 3..4, Y in 4..6 ?
```

4. Írd meg a következő predikátumot a `library(clpfd)` segítségével! Ne használj választási pontokat (spekulatív diszjunkciót)!

`profilja(L, P)`: Az L n elemű lista profilja a P n-1 elemű lista, ha $P_i = sign(L_{i+1} - L_i)$, i=1, ..., n-1. Azaz, ha L első két eleme a és b, akkor P első elemének értéke 1, ha $a<b$; -1, ha $a>b$; és 0, ha $a=b$, stb.

Példák:

```
| ?- length(L, 4), domain(L, 1, 3), profilja(L, [1,1,1]).
no
| ?- length(L, 4), domain(L, 1, 3), profilja(L, [1,1,0]).
L = [1,2,3,3] ? ;
no
```

3.2. Az 1997. évi első feladatsor

1. Határozd meg a változók tartományának változását az alábbi célsorozat végrehajtása során! Lépésenként írd le, hogy az egyes korlátokból milyen démonok keletkeznek, ezek mikor ébrednek fel milyen szűkítést végezve, és mikor fejezik be működésüket!

```
?- domain([X,Y], 0, 100), X+Y #>= 9, element(X, [1,2,6,24,120], Y), X-Y #> 3.
```

2. Írj egy `library(clpfd)` programot az alábbi feladat megoldására!

`joszam(X)`: Az `X` egész szám kétjegyű, négyzete háromjegyű és a négyzet első két jegye forított sorrendben leírva az `X` számot adja.

3. Definiáld FD-predikátummal (indexikálisokkal) az `'x-y>=10'(X, Y)` constraintet, amely nevének megfelelő jelentésű! Írd meg az FD-predikátum mind a négy klózát!

4. Írd meg a mastermind játékkal kapcsolatos következő predikátumot a `library(clpfd)` segítségével! Ne használj választási pontokat (spekulatív diszjunkciót)!

`feketek(Rk, Tk, N, F)`: Az `Rk` (rejtettek) változó-lista és a `Tk` (tippek) egész-lista azonos hosszúak. `N` és `F` konkrét egész számok. `Rk` minden eleme az `1..N` tartományba esik. A predikátum jelentése: Az `F` szám (fekete találatok) nem más mint az `Rk` lista azon elemeinek száma, amelyek értéke megegyezik a `Tk` lista azonos sorszámú elemével.

Példa:

```
?- feketek([A,B,C], [1,2,3], 3, 0).

A in {2,3}, B in {1,3}, C in {1,2}
```

3.3. Az 1997. évi második feladatsor

1. Határozd meg a változók tartományának változását az alábbi célsorozat végrehajtása során! Lépésenként írd le, hogy az egyes korlátokból milyen démonok keletkeznek, ezek mikor ébrednek fel milyen szűkítést végezve, és mikor fejezik be működésüket!

```
?- domain([X,Y], 1, 10), X+Y #=< 10,
   element(X, [1,1,4,4,9,9], Y), X #> Y.
```

2. Írj egy `library(clpfd)` programot az alábbi feladat megoldására!

`joszam(X)`: Az `X` egész szám kétjegyű, négyzete háromjegyű és a négyzet első két jegye az `X` számnál 2-vel nagyobb szám.

3. Definiáld FD-predikátummal (indexikálisokkal) az `'x+y=<z'(X, Y, Z)` constraintet, amely nevének megfelelő jelentésű, és intervallum-konzisztenciát ill. -levezethetőséget biztosít! Írd meg az FD-predikátum mind a négy klózát!

4. Írd meg a következő eljárást a `library(clpfd)` segítségével! Ne használj választási pontokat (spekulatív diszjunkciót)!

`jo(N, L, T)`: Az `L` lista `N` hosszú és elemei az `1..N` intervallumból való különböző számok, továbbá az `I`-edik pozíción nem állhat az `I` szám. `T` az `abs(L[I]-I)` eltérések összege, `I= 1..N`. Adott `N` esetén `L` egy olyan lista, amelyre `T` minimális.

3.4. Az 1997. évi harmadik feladatsor

1. Határozd meg a változók tartományának változását az alábbi célsorozat végrehajtása során! Lépésenként írd le, hogy az egyes korlátokból milyen démonok keletkeznek, ezek mikor ébrednek fel milyen szűkítést végezve, és mikor fejezik be működésüket!

```
?- domain([X,Y], 1, 10), X+Y #=< 10,
   relation(X, [4-{2},6-{2,3},8-{2,4},9-{3},10-{2,5}], Y),
   X-Y #> 3, indomain(Y).
```

2. Írj egy `library(clpfd)` programot az alábbi feladat megoldására!

Az `egyszerusitheto(X,Y)` eljárás sorolja fel a következő tulajdonságokkal bíró `(X,Y)` számpárokat:

- `X` és `Y` tizes számrendszerben kétjegyű természetes számok
- `X` tizes számrendszerbeli alakja AB
- `Y` tizes számrendszerbeli alakja BC
- Az `X/Y` és A/C törtek (végtelen pontossággal) megegyeznek
- A, B és C páronként különböző számjegyek

3. Definiáld FD-predikátummal (indexikálisokkal) az `'abs(x-y)=<10'(X, Y)` constraintet, amely nevének megfelelő jelentésű, és intervallum-konzisztenciát ill -levezethetőséget biztosít! Írd meg az FD-predikátum mind a négy klózát!

4. Írd meg a következő constraintet a `library(clpfd)` segítségével! Ne használj választási pontokat (spekulatív diszjunkciót)!

`kiegyensulyozott(L)`: Az `L` listában azon pozíciók száma, amelyeken az utána levőnél nagyobb szám áll megegyezik azon pozíciók számával, amelyeken az utána levőnél kisebb szám áll.

Példa:

```
| ?- L=[A,B,C], domain(L, 1, 3), A+B #=4, kiegyensulyozott(L),
     labeling([],L), write(L), nl, fail.
[1,3,1]
[1,3,2]
[2,2,2]
[3,1,2]
[3,1,3]

no
| ?-
```