Nagyhatékonyságú Deklaratív Programozás — ZH minta-feladatok

Az alábbi 1.-4. feladatsorok mindegyikéből 1-1 feladat szerepel majd a ZH-n:

1. FD-predikátumok
2. Globális korlátok
3. CLPFD feladatok
4. Egyéb feladatok: a CLPQ, CHR, Mercury témakörök egyike

1. FD-predikátumok

1.1. 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.

1.2. Definiáld az `'x>0\/y>0'(X, Y)` FD-predikátumot, amely nevének megfelelő jelentésű, és lehetőleg minél erősebb szűkítést biztosít! Csak a `+:` nyakjelű klózt kell megírnod!
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 ?
```

1.3. Definiáld FD-predikátummal (indexikálisokkal) az `'x=<y=<2x'(X, Y)` korlátot, amely nevének megfelelő jelentésű, és lehetőleg minél erősebb szűkítést/levezethetőséget biztosít! Csak a `+:` és a `-?` nyakjelű klózokat kell megírnod. Mielőtt megirnád az indexikálisokat, írd le szövegesen (kommentben), hogy milyen szűkítést vársz tőlük, illetve milyen levezethetőségi feltételt akarsz velük kifejezni.
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,  Y in (inf..2)\/(9..sup), 'x=<y=<2x'(X,Y) #<=> B.
                                B = 0, X in 3..4, Y in(inf..2)\/(9..sup) ?
```

2. Globális korlátok

2.1. 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
```

2.2. 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), 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
```

2.3. Nevezzünk egy `L` listát `[N,M]`-metszőnek, ha `L`-nek legalább egy olyan eleme van, amely az `[N,M]` zárt számintervallumba esik.

Globális korlátként írj meg egy `metszo(L, N, M)` predikátumot, amely azt fejezi ki, hogy az `L` lista `[N,M]`-metsző! `L` korlát-változók listája, `N` és `M` adott egész számok. Figyelj arra, hogy a globális korlát kilépjen, mihelyst a tárból következik az `[N,M]`-metszési feltétel igaz vagy hamis volta (mint az alábbi példák mindegyikében)!

Példák:

```
| ?- _L = [_A,B,_C], _A #<2, _C in 10..20, metszo(_L, 3, 5).
        B in 3..5 ? ; no
| ?- _L = [1,_B,C], _B #< 3, C in 5..20, metszo(_L, 3, 5).
        C = 5 ? ; no
| ?- _L = [_,_A,_], _A in 4..5, metszo(_L, 3, 5).
        true ? ; no
| ?- _L = [1,_B,_C], _B #< 3, _C #> 5, metszo(_L, 3, 5).
        no
```

3. CLPFD feladatok

3.1. 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
```

3.2. 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([0,1,1,0,0,0,1,1,1], 9, [2,3])` igaz.

Az egyesekből álló folytonos szakaszok között tetszőleges számú (de legalább 1) 0 kell szerepeljen, a lista elején és végén is állhat nullákból álló sorozat, de ez nem kötelező.

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] ?
```

3.3. 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}
```

4. Egyéb feladatok

4.1. 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] ?
```

4.2. 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).
```

4.3. Tekintsd az alábbi `a/4` Mercury predikátumot!

```
a(A, B, C, ABC) :-
        b(A, BC, ABC),
        b(B, C, BC).
```

Tegyük fel, hogy az `a/4`-ben felhasznált `b/3` predikátumra az alábbi deklarációkat adtuk meg:

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

Add meg az `a` predikátum-deklarációját (azaz az argumentumai típusát) és az (`in` / `out` módokat használó) mind a 9 lehetséges predikátummód-deklarációját, determinizmusjelzéssel együtt! 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.