

CLPFD segéd eljárások

Statisztika

- `fd_statistics(Kulcs, Érték)`: A Kulcs-hoz tartozó számláló Érték-ét kiadja és lenullázza. Lehetséges kulcsok és számlált események:
 - `constraints` — korlát létrehozása;
 - `resumptions` — korlát felélesztése;
 - `entailments` — korlát (vagy negáltja) levezethetővé válásának észlelése;
 - `prunings` — tartomány szűkítése;
 - `backtracks` — a tár ellentmondásossá válása (Prolog meghiúsulások nem számítanak).
- `fd_statistics`: az összes számláló állását kiírja és lenullázza őket.

```
% Az N- királynő feladat összes megoldása Ss, Lab címkézéssel való
% végrehajtása Time msec-ig tart és Btrks FD visszalépést igényel.
run_queens(Lab, N, Ss, Time, Btrks) :-
    fd_statistics(backtracks, _), statistics(runtime, _),
    findall(Q, queens(Lab, N, Q), Ss),
    statistics(runtime, [_Time]),
    fd_statistics(backtracks, Btrks).
```

Válaszok formája (a még le nem futott, alvó korlátok kiírása a válaszban)

- `clpfd:full_answer`: ez egy dinamikus kampó eljárás. Alaphelyzetben nincs egy klóza sem, tehát nem sikerül. Ez esetben a rendszer egy kérdésre való válaszláskor csak a kérdésben előforduló változók tartományát írja ki, az alvó korlátokat nem. Ha felveszünk egy ilyen eljárást és az sikeresen fut le, akkor a válaszban az összes változó mellett kiírja még a le nem futott összes korlátot is.

```
| ?- domain([X,Y], 1, 10), X+Y#=5. => X in 1..4, Y in 1..4 ?
| ?- assert(clpfd:full_answer). => yes
| ?- domain([X,Y], 1, 10), X+Y#=5. => clpfd:'t+u=c'(X,Y,5),
|                                     X in 1..4, Y in 1..4 ?
| ?- X+Y #= Z #<=> B. => clpfd:'t=u IND'(Z,_A)#<=>B,
|                                     clpfd:'x+y=t'(X,Y,_A), B in 0..1, ...
| ?- retract(clpfd:full_answer). => yes
| ?- X+Y #= Z #<=> B. => B in 0..1, ...
```

77

CLPFD segéd eljárások (folyt.)

FD változók belső jellemzői

- Az FD változókról a könyvtár által tárolt információk lekérdezhetők.
- Ezek felhasználhatók a címkézésben, globális korlátok írásában ill. nyomkövetésben.
- Vigyázat!** Félreértés veszélye! Minden más használat nagy eséllyel hibás.

FD változók felismerése

- `fd_var(V)`: V egy a clpfd könyvtár által ismert változó.

Tartományok pillanatnyi jellemzőinek lekérdezése

- `fd_min(X, Min)`: A Min paramétert egyesíti az X változó tartományának alsó határával (ez egy szám vagy inf lehet).
- `fd_max(X, Max)`: Max az X felső határa (szám vagy sup).
- `fd_size(X, Size)`: Size az X tartományának számossága (szám vagy sup).
- `fd_dom(X, Range)`: Range az X változó tartománya, *KonstansTartomány* formában
- `fd_set(X, Set)`: Set az X tartománya ún. FD-halmaz formában.
- `fd_degree(X, D)`: D az X-hez kapcsolódó korlátok száma.

Példák

```
| ?- X in (1..5)\{9}, fd_min(X, Min), fd_max(X, Max),
|     fd_size(X, Size).
|     Min = 1, Max = 9, Size = 6, X in(1..5)\{9} ?
| ?- X in (1..9)\(6..8), fd_dom(X, Dom), fd_set(X, Set).
|     Dom = (1..5)\{9}, Set = [[1|5],[9|9]], X in ... ?
| ?- queens_nolab(8, [X|_]), fd_degree(X, Deg).
|     Deg = 21, X in 1..8 ? % 21 = 7*3
```

78

FD-halmazok

Az FD-halmaz fogalma, alapműveletei

- Az FD-halmaz formátum a tartományok belső ábrázolási formája.
- Absztrakt adattípusként használandó, alapműveletei:
 - `is_fdset(S)`: S egy korrekt FD-halmaz.
 - `empty_fdset(S)`: S az üres FD-halmaz.
 - `fdset_parts(S, Min, Max, Rest)`: Az S FD-halmaz áll egy Min..Max kezdő intervallumból és egy Rest maradék FD-halmazból, ahol Rest minden eleme nagyobb Max+1-nél. Egyaránt használható FD-halmaz szétszedésére és építésére (ez utóbbi most éppen hibás).

```
| ?- X in (1..9) /\ (6..8), fd_set(X, _S),
|     fdset_parts(_S, Min1, Max1, _).
|     Min1 = 1,
|     Max1 = 5,
|     X in(1..5)\{9} ?
```

- Az FD-halmaz tényleges ábrázolása: `[Alsó|Felső]` alakú szeparált zárt intervallumok rendezett listája. (A '(_,_)' struktúra memóriaigénye 33%-kal kevesebb mint bármely más 'f(_,_)' struktúráé.)

```
| ?- X in (1..9) /\ (6..8), fd_set(X, S).
|     S = [[1|5],[9|9]],
|     X in(1..5)\{9} ?
```

- FD-halmaz is használató szűkítésre:

- `X in_set Set`: Az X változót a Set FD-halmazzal szűkíti.
- Vigyázat!** Ha a korlát-felvételi fázisban egy változó tartományát egy másik tartományának függvényében szűkítünk, ezzel nem érhetünk el „démoni” szűkítő hatást, hiszen ez a szűkítés csak *egyszer* fut le. Az `in_set` eljárást csak globális korlátok ill. testreszabott címkézés megvalósítására célszerű használni.

79

FD-halmazok (folyt.)

FD-halmazokat kezelő további eljárások

- `fdset_singleton(Set, E1t)`: Set az egyetlen E1t-ből áll.
- `fdset_interval(Set, Min, Max)`: Set a Min..Max intervallum (oda-vissza használható).
- `fdset_union([3,2], fdset_intersection([3,2])`: Két halmaz ill. egy listában megadott halmazok úniója és metszete.
- `fdset_complement/2`: Egy halmaz komplementje.
- `fdset_member(E1t, Set)`: E1t eleme a Set FD-halmaznak.
- `list_to_fdset(List, Set), fdset_to_list(Set, List)`: számlista átalakítása halmazzá, és fordítva.
- `range_to_fdset(Range, Set), fdset_to_range(Set, Range)`: Konstans tartomány átalakítása halmazzá és viszont.

Példa

```
| ?- list_to_fdset([2,3,5,7], _FS1),
|     fdset_complement(_FS1, _FS2),
|     % _FS2 ⇔ {2,3,5,7}
|     fdset_interval(_FS3, 0, sup),
|     % _FS3 ⇔ 0..sup
|     fdset_intersection(_FS2, _FS3, _FS4),
|     % _FS4 ⇔ (0..sup)\{2,3,5,7}
|     fdset_to_range(_FS4, Range),
|     X in_set _FS4.
```

```
Range = (0..1)\{4}\{6}\(8..sup),
X in(0..1)\{4}\{6}\(8..sup) ?
```

80

Címkézési (keresési) stratégiák

CSP programok szerkezete (ismétlés!)

- változók és tartományaik megadása,
- korlátok felvétele (lehetőleg választási pontok létrehozása nélkül!),
- címkézés (keresés).

A címkézési fázis feladata

- Adott változók egy halmaza,
- ezeket a tartományaik által megengedett értékekre szisztematikusan be kell helyettesíteni
- (miközben a korlátok fel-felébrednek, és visszalépést okoznak a nem megengedett állapotokban).
- Mіндеzt a lehető leggyorsabban, a lehető legkevesebb visszalépéssel kell megoldani.

A keresés célja lehet

- egyetlen** (tetszőleges) megoldás előállítás,
- az **összes** megoldás előállítása,
- a valamilyen szempontból **legjobb** megoldás előállítása.

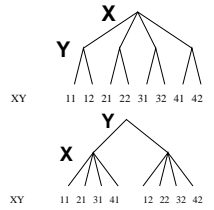
A keresési stratégia paraméterezési lehetőségei

- Milyen **sorrendben** kezeljük az egyes változókat?
- Milyen **választási pontot** hozunk létre?
- Milyen **irányban** járjuk be a változó tartományát?

81

Hogyan függ a keresési tér a változó-sorrendtől?

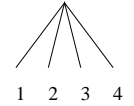
- | ?- X in 1..4, Y in 1..2, indomain(X), indomain(Y).
- | ?- X in 1..4, Y in 1..2, indomain(Y), indomain(X).



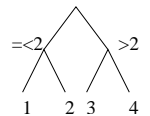
- A **first-fail** elv: a kisebb tartományú változót előbb címkézzük — kevesebb választási pont, remélhetően kisebb keresési tér.
- Példa feladatspecifikus sorrendre: az N királynő feladatban érdemes a középső sorokba tenni le először a királynőket, mert ezek a többi változó tartományát jobban megsűrik, mint a szélsőkbe tették.

Milyen szerkezetű keresési tereket hozhatunk létre?

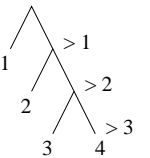
- felsorolás: | ?- X in 1..4, labeling([enum], [X]).



- kettévágás: | ?- X in 1..4, labeling([bisect], [X]).



- lépegetés: | ?- X in 1..4, labeling([step], [X]).



82

Címkéző eljárások; labeling/2

A címkézés alap-eljárása: labeling(Opciók, VáltozóLista)

A VáltozóLista minden elemét behelyettesíti, az Opciók lista által előírt módon. Az alábbi csoportok mindegyikéből legfeljebb egy opció szerepelhet. Ha az első négy csoport valamelyikéből nem szerepel opció, akkor a *dőlt betűvel* szedett alapértelmezés lép életbe.

- a változó kiválasztása: *leftmost*, *min*, *max*, *ff*, *ffc*, *variable*(Sel)
- a választási pont fajtája: *step*, *enum*, *bisect*, *value*(Enum)
- a bejárési irány: *up*, *down*
- a keresett megoldások: *all*, *minimize*(X), *maximize*(X)
- a gyűjtendő statisztikai adat: *assumptions*(A)
- a balszélső ágtól való eltérés korlátozása: *discrepancy*(D)

A címkézés menete

- Ha a VáltozóLista üres, akkor a címkézés sikeresen véget ér. Egyébként kiválasztunk belőle egy X elemet az 1. csoportbeli opció által előírt módon.
- Ha X már behelyettesített, akkor a Változólistából elhagyjuk, és folytatjuk az **a.** pontnál.
- Egyébként az X változó tartományát felosztjuk két vagy több diszjunkt részre a 2. csoportbeli opció szerint (kivéve *value*(Enum) esetén, amikor is azonnal az **e.** pontra megyünk).
- A tartományokat elrendezzük a 3. csoportbeli opció szerint.
- Létrehozunk egy választási pontot, amelynek ágain sorra leszűkítjük az X változót a kiválasztott tartományokra.
- Minden egyes ágon az X szűkítése értelemszerűen kiváltja a rá vonatkozó korlátok felébredését. Ha ez meghiúsulást okoz, akkor visszalépünk az **e.** pontra és ott a következő ágon folytatjuk.
- Ha X most már behelyettesített, akkor elhagyjuk a Változólistából. Ezután mindenképpen folytatjuk az **a.** pontnál.
- A fenti folyamat során értelemszerűen figyelembe vesszük a 4.-6. csoportbeli opciók előírásait is.

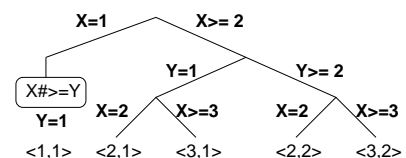
83

A címkézés menete — példa

- A példa:
X in 1..3, Y in 1..2, X#>=Y, labeling([min], [X,Y]).
- A *min* opció a legkisebb alsó határu változó kiválasztását írja elő.

```
| ?- fdbg_assign_name(X, x), fdbg_assign_name(Y, y),
X in 1..3, Y in 1..2, X#>=Y, fdbg_on,
labeling([min], [X,Y]).
% The clp(fd) debugger is switched on
Labeling [1, <x>]: starting in range 1..3.
Labeling [1, <x>]: step: <x> = 1
<y>#<=1 y = 1..2 -> {1} Constraint exited.
X = 1, Y = 1 ? ;
Labeling [1, <x>]: step: <x> >= 2
<y>#<=<x> y = 1..2, x = 2..3 Constraint exited.
Labeling [6, <y>]: starting in range 1..2.
Labeling [6, <y>]: step: <y> = 1
Labeling [8, <x>]: starting in range 2..3.
Labeling [8, <x>]: step: <x> = 2
X = 2, Y = 1 ? ;
Labeling [8, <x>]: step: <x> >= 3
X = 3, Y = 1 ? ;
Labeling [8, <x>]: failed.
Labeling [6, <y>]: step: <y> >= 2
Labeling [12, <x>]: starting in range 2..3.
Labeling [12, <x>]: step: <x> = 2
X = 2, Y = 2 ? ;
Labeling [12, <x>]: step: <x> >= 3
X = 3, Y = 2 ? ;
Labeling [12, <x>]: failed.
Labeling [6, <y>]: failed.
Labeling [1, <x>]: failed.
```

A keresési fa



84

Címkézési opciók

A címkézendő változó

A következő címkézendő változó kiválasztási szempontjai (ahol több szempont van, a későbbi csak akkor számít, ha a megelőzők egyenlők):

- `leftmost` — legbaloldali (alapértelmezés);
- `min` — a legkisebb alsó határú; a legbaloldali;
- `max` — a legnagyobb felső határú; a legbaloldali;
- `ff` — („first-fail” elv): a legkisebb tartományú; a legbaloldali;
- `ffc` — a legkisebb tartományú; a legtöbb korlátban előforduló; a legbaloldali;
- `variable(Sel)` — (meta-opció) `Sel` egy felhasználói eljárás, amely kiválasztja a következő címkézendő változót (lásd később).

A választás fajtája

A kiválasztott `X` változó tartományát a következőképpen bonthatjuk fel:

- `step` — `X #= B` és `X #\= B` közötti választás, ahol `B` az `X` tartományának alsó vagy felső határa (alapértelmezés);
- `enum` — többszörös választás `X` lehetséges értékei közül;
- `bisect` — `X #< M` és `X #>= M` közötti választás, ahol `M` az `X` tartományának középső eleme ($M = (\min(X) + \max(X))/2$);
- `value(Enum)` — (meta-opció) `Enum` egy eljárás, amelynek az a feladata, hogy leszűkítse `X` tartományát (lásd később).

A bejárás iránya

A tartomány bejárás iránya lehet:

- `up` — alulról felfelé (alapértelmezés);
- `down` — felülről lefelé.

85

Címkézési opciók (folyt.)

A keresett megoldások

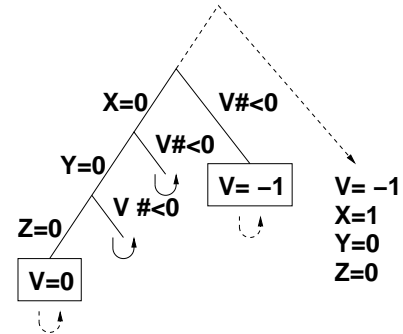
- `all` — visszalépéssel az összes megoldást felsorolja (alapértelmezés);
- `minimize(X)` ill. `maximize(X)` — egy, az `X`-re minimális ill. maximális értéket eredményező megoldást keres, branch-and-bound algoritmussal.

Példa szélsőérték keresésére

```
| ?- _L=[X,Y,Z], domain(_L, 0, 1),
      V#=Y+Z-X, labeling([minimize(V)], _L).

V = -1, X = 1, Y = 0, Z = 0 ? ;
no
```

A keresési fa a branch-and-bound algoritmussal



86

Címkézési opciók (folyt.)

Egyéb opciók

- **Statistika:** `assumptions(K)` — egyesíti `K`-t a sikeres megoldáshoz vezető ágon levő változó-kiválasztások számával (ami lényegében a keresési fában a megoldáshoz vezető út hossza).
- A heurisztikától való eltérés korlátozása: `discrepancy(D)` (`D` adott szám)— csak olyan megoldásokat kérünk figyelembe venni, amelyekhez a keresési fában úgy jutunk el, hogy a legfeljebb `D`-szer választunk egy nem-legbaloldali ágat a választási pontokban. (Szemléletesen: a fa gyökerétől a megoldásig haladva legfeljebb `D`-szer kell megadni a jobbkez-szabály szerinti elsőbbséget.) Az opció háttere az LDS (Limited Discrepancy Search) keresési módszer. Ebben feltételezzük, hogy a legbaloldali választások képviselik azt a heurisztikát, amivel nagy valószínűséggel eljuthatunk egy megoldáshoz. Mivel a heurisztika nem teljesen tökéletes, ezért valamennyi eltérést megengedünk, de az össz-eltérés-mennyiséget korlátozzuk.

Példák (vö. a 82. lapon levő keresési fákkal):

```
assumptions(Select, As) :-
    X in 1..4,
    findall(A, labeling([Select,
                        assumptions(A)], [X]), As).

lds(Select, D, Xs) :-
    X in 1..4,
    findall(X, labeling([Select,
                        discrepancy(D)], [X]), Xs).

| ?- assumptions(enum, As).           As = [1,1,1,1]
| ?- assumptions(bisect, As).        As = [2,2,2,2]
| ?- assumptions(step, As).         As = [1,2,3,3]

| ?- lds(enum, 1, Xs).              Xs = [1,2,3,4]
| ?- lds(bisect, 1, Xs).           Xs = [1,2,3]
| ?- lds(step, 1, Xs).             Xs = [1,2]
```

87

A címkézés testreszabása

labeling/2 — a variable(Sel) meta-opció

- `variable(Sel)` — `Sel` egy eljárás, amely kiválasztja a következő címkézendő változót. `Sel(Vars, Selected, Rest)` alakban hívja meg a rendszer, ahol `Vars` a még címkézendő változók/számok listája.
- `Sel`-nek determinisztikusan sikerülnie kell egyesítve `Selected`-et a címkézendő változóval és `Rest`-et a maradékkal.
- `Sel` egy tetszőleges meghívható kifejezés lehet (`callable`, azaz név vagy struktúra). A három argumentumot a rendszer fűzi `Sel` argumentumlistájának végére.
- Például: ha a `Sel` opcióként a `mod:sel(Param)` kifejezést adjuk meg, akkor a rendszer a `mod:sel(Param, Vars, Selected, Rest)` eljáráshívást hajtja majd végre.

Példa a variable opció használatára

```
% A Vars-beli változók között Sel a Hol-adik,
% Rest a maradék.
valaszt(Hol, Vars, Sel, Rest) :-
    szur(Vars, Szurtek),
    length(Szurtek, Len), N is integer(Hol*Len),
    nth0(N, Szurtek, Sel, Rest).

% szur(Vk, SzK): A Vk-ban levő változók listája SzK.
szur([], []).
szur([V|Vk], SzK) :- nonvar(V), !, szur(Vk, SzK).
szur([V|Vk], [V|SzK]) :- szur(Vk, SzK).

queens([], 8, Qs).           → Qs = [1,5,8,6,3,7,2,4]
queens([variable(valaszt(0.5))], 8, Qs)
    → Qs = [7,2,6,3,1,4,8,5]
queens([variable(valaszt(0.7))], 8, Qs)
    → Qs = [5,7,2,6,3,1,4,8]
```

88

A címkézés hatékonysága

A címkézés testreszabása (folyt.)

labeling/2 — a value(Enum) meta-opció

- value(Enum) — Enum egy eljárás, amelynek az a feladata, hogy leszűkítse X tartományát. Az eljárást a rendszer Enum(X , Rest, BB0, BB) alakban hívja meg, ahol $[X|Rest]$ a még címkézendő változók listája.
- Enum-nak nemdeterminisztikusan le kell szűkítenie X tartományát az összes lehetséges módon, vö. a címkézés menetének leírását a 83. lapon.
- Az első választásnál meg kell hívnia az first_bound(BB0, BB), a későbbieknél a later_bound(BB0, BB) eljárást, a BB ill. LDS keresési algoritmusok kiszolgálására.
- Enum-nak egy meghívható kifejezésnek kell lennie. A négy argumentumot a rendszer fűzi Enum argumentumlistájának a végére.

Példa: belülről kifelé való érték-felsorolás

```
midout(X, _Rest, BB0, BB) :-
    fd_size(X, Size),
    Mid is (Size+1)//2,
    fd_set(X, Set),
    fdset_to_list(Set, L),
    nth(Mid, L, MidElem),
    ( first_bound(BB0, BB), X = MidElem
    ; later_bound(BB0, BB), X #\= MidElem
    ).

| ?- X in {1,3,12,19,120},
    labeling([value(midout)], [X]).
X = 12 ? ;
X = 3 ? ;
X = 19 ? ;
X = 1 ? ;
X = 120 ? ;
no
```

89

A korábbi queens eljárás megoldásait különböző címkézési opciókkal, különböző méretű táblákra, 600 MHz Pentium III gépen.

(Jelölés: midvar \equiv variable(valaszt(0.5)).)

Összes megoldás keresése

méret	n=8		n=10		n=12	
megoldások száma	92		724		14200	
címkézés	sec	btrk	sec	btrk	sec	btrk
[step]	0.23	324	4.09	5942	100.1	131K
[enum]	0.22	324	4.00	5942	97.3	131K
[bisect]	0.22	324	4.11	5942	100.4	131K
[enum,min]	0.28	462	5.04	8397	134.3	202K
[enum,max]	0.27	462	5.04	8397	134.6	202K
[enum,ff]	0.21	292	3.53	4992	79.9	101K
[enum,ffc]	0.22	292	3.60	4992	81.5	101K
[enum,midvar]	0.20	286	3.32	4560	74.3	88K

Első megoldás keresése

méret	n=16		n=18		n=20	
címkézés	sec	btrk	sec	btrk	sec	btrk
[enum]	1.79	1833	7.97	7436	42.35	37320
[enum,min]	2.12	2095	3.39	2595	5.33	3559
[enum,max]	2.58	3182	12.28	13917	75.00	83374
[enum,ff]	0.06	7	0.08	11	0.15	33
[enum,ffc]	0.05	7	0.06	11	0.15	33
[enum,midvar]	0.11	69	0.12	57	0.55	461
[value(midout),ffc]	0.05	5	0.04	1	0.11	7

90

További címkéző eljárások

indomain(X) — ekvivalens a labeling([enum], [X]) hívással.

minimize(Cél, X) ill. maximize(Cél, X)

A Cél *ismélt hívásával* megkeresi az X változó minimális ill. maximális értékét.

A minimize/2 eljárás definíciója

```
minimize(Goal, Var) :-
    findall(Goal-Var, (Goal -> true), [Best1-UB1]),
    minimize(Goal, Var, Best1, UB1).

% minimize(Goal, Var, BestGoal, UB): Var is the minimal
% value < UB allowed by Goal, or, failing that,
% Goal = BestGoal.
minimize(Goal, Var, _, UB) :- var(UB), !,
    % hibajelzés.
minimize(Goal, Var, _, UB) :-
    Var #< UB,
    findall(Goal-Var, (Goal -> true), [Best1-UB1]), !,
    minimize(Goal, Var, Best1, UB1).
minimize(Goal, Var, Goal, Var).
```

Példa a minimize/2 használatára

```
p(L, V) :-
    L = [X,Y,Z], domain(L, 0, 1),
    V #= Y+Z-X, labeling([], L).

| ?- spy([p/2, minimize/4, call/1]).
| ?- minimize1(p(L, V), V).
+ 1 1 Call: p(L,V) ? z
+ 1 1 Exit: p([0,0,0],0) ? z
+ 2 1 Call: minimize(p(L,V),V,p([0,0,0],0),0) ? z
+ 3 2 Call: call(user:(V#<0)) ? z
+ 3 2 Exit: call(user:(V#<0)) ? z
+ 4 2 Call: p(L,V) ? z
+ 4 2 Exit: p([1,0,0],-1) ? z
+ 5 2 Call: minimize(p(L,V),V,p([1,0,0],-1),-1) ? z
+ 6 3 Call: call(user:(V#<-1)) ? z
+ 6 3 Exit: call(user:(V#<-1)) ? z
+ 7 3 Call: p(L,V) ? z
+ 7 3 Fail: p(L,V) ? z
+ 5 2 Exit: minimize(p([1,0,0],-1),-1,p([1,0,0],-1),-1) ? z
+ 2 1 Exit: minimize(p([1,0,0],-1),-1,p([0,0,0],0),0) ? z
L = [1,0,0], V = -1 ?
```

91

2. kis házi feladat: számkeresztrejtvény

A feladat

- Adott egy keresztrejtvény, amelyek egyes kockáiba $1..Max$ számokat kell elhelyezni (szokásosan $Max = 9$).
- A vízszintes és függőleges „szavak” meghatározásaként a benne levő számok összege van megadva.
- Egy szóban levő betűk (kockák) mind különböző értékkel kell bírjanak.

A keresztrejtvény Prolog ábrázolása:

- listák listájaként megadott mátrix;
- a fekete kockák helyén $F \setminus V$ alakú struktúrák vannak, ahol F és V az adott kockát követő függőleges ill. vízszintes szó összege, vagy x , ha nincs ott szó;
- a kitöltendő fehér kockákat (különböző) változók jelzik.

A megírandó Prolog eljárás és használata

```
% szamker(SzK, Max): SzK az 1..Max számokkal
% helyesen kitöltött számkeresztrejtvény.
% Megjegyzés: egyes sorban/oszlopban közepén
% is lehet 'x'!
```

```
pelda(mini, [[x\ x,11\ x,21\ x, 8\ x],
             [x\24,  ,  ,  ,  ],
             [x\10,  ,  ,  ,  ],
             [x\6,  ,  ,  , x\x]], 9).
```

		11	21	8
24	8	9	7	
10	2	7	1	
6	1	5		

```
| ?- pelda(mini, SzK, _Max), szamker(SzK, _Max).
SzK = [[x\ x, 11\ x, 21\ x, 8\ x],
        [x\24, 8, 9, 7 ],
        [x\10, 2, 7, 1 ],
        [x\6, 1, 5, x\ x]] ? ; no
```

92