

Nagyhatékonyságú Deklaratív Programozás

GYAKORLÓ FELADATOK

Korutinszervezési lehetőségek

Ha a feladat szövege másként nem rendelkezik, írd meg az alábbi fejkomenteknek megfelelő Prolog eljárásokat! Törekedj minél hatékonyabb megoldásra! Használd a SICStus korutinszervezési lehetőségeit!

1. Vizsgáld meg az alábbi Prolog programot, amely az N -vezér feladatot oldja meg „generate and test” módszerrel! Írj egy hatékonyabb változatot, `queens_c/2` néven, a korutinszervezési lehetőségek felhasználásával!

```
% queens(+N, ?Q): Q is a *safe* placement of queens, i.e. no two queens attack each other.
```

```
queens(N, Qs) :-  
    length(Qs, N), place_queens(Qs, N), safe(Qs).
```

```
% List Qs, of length N, describes a placement of queens on a chessboard of
```

```
% size N*N: if Q[i] = j, then a queen is placed in row i, column j.
```

```
place_queens([], _).
```

```
place_queens([Q|Qs], N) :-
```

```
    between(1, N, Q), place_queens(Qs, N).
```

```
% safe(Qs): List Qs describes a safe placement of queens.
```

```
safe([]).
```

```
safe([Q|Qs]) :-
```

```
    no_attack(Qs, Q, 1), safe(Qs).
```

```
% no_attack(Qs, Q, I): A queen placed in row i, position Q does not attack
```

```
% any of the queens Qs[1], Qs[2], etc, placed in rows i+I, i+I+1, etc.
```

```
no_attack([], _, _).
```

```
no_attack([X|Xs], Y, I) :-
```

```
    no_threat(X, Y, I), J is I+1, no_attack(Xs, Y, J).
```

```
% Queens placed in columns X and Y in rows of distance I do not attack each
```

```
% other. A bit more formally: queens placed in row i, column X and in row
```

```
% i+I, column Y do not attack each other.
```

```
no_threat(X, Y, I) :-
```

```
    Y =\= X, Y =\= X-I, Y =\= X+I.
```

```
% between(+M, +N, ?X): X is an integer such that M <= X <= N, where M and N are integers.
```

```
between(M, N, M) :-
```

```
    M <= N.
```

```
between(M, N, I) :-
```

```
    M < N, M1 is M+1, between(M1, N, I).
```

2. Adott szám felbontása három relatív prím szám összegére. Ötlet: A SICStus Prolog tartalmazza a `gcd` (LNKO) kétargumentumú aritmetikai függvényt, pl. `?- X is gcd(24,45). --> X = 3 ? ; no`

```
% felbontas(+X, -A, -B, -C):
```

```
% X, A, B, C pozitív egész számok
```

```
% X = A + B + C
```

```
% 1 < A < B < C
```

```
% A, B, C számok páronként relatív prímek
```

3. Az egyszerűsített Hamming probléma

```
% hamming(+N, ?H): A H lista az első N olyan természetes számot tartalmazza növekvő sorrendben, amelyek
```

```
% prímtényezői között csak a 2 és 3 számok szerepelnek. (N > 1)
```

Segédeljárások:

```
% times(L1, M, L): Az L lista az L1 lista elemeinek M-szerese.
```

```
% merge(L1, L2, L): Az L lista az L1 és L2 rendezett listák rendezett, ismétlődésmentes összefésülése
```

```
% prefix(N, L1, L): Az L1 lista első N eleme L.
```

4. Diszjunkciós egyenletek megoldása. Az alábbi eljárások ne hozzanak létre választási pontot, viszont ha az argumentumok kellően behelyettesítettek, akkor az összes lehetséges következtetést végezzék el.

```
% +(A, B, C): A és B diszjunkciója C (A, B, C 0-1 értékű változók)
```

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

```
% sat(Kif): Kif egy Kif1 == Kif2 alakú kifejezés, amelyben a két oldalon levő kifejezések változókból
```

```
% illetve 0 és 1 konstansokból a + operátor segítségével épített kifejezések.
```