```
% GRUPO II - Exame PLOG 2010/2011
% FEUP, 31/01/2011
% Daniel Moura (daniel.moura@fe.up.pt)


:-use_module(library(clpfd)).


p6(Vars) :-
    Vars = [X,Y,Z],

    X in {2}\/{3}\/{5}\/{7}, % limita o dominio aos valores possiveis
    Y in 12..97, % ou: Y in 0..100
    Z in {1}\/{3}\/{5}\/{7}\/{9}, % mais eficiente que: Z mod 2 #= 1
    Y mod 10 #= X,
    X+Y+Z #= A*A,
    X #> Z,
    all_distinct(Vars), % ou: all_different(Vars)  ou: X\=Y,Y\=Z
    labeling([],Vars).


% ----------------------------------------


p7a(Vars) :-
    Vars = [_A,_B,_C,_D,_E,_F,_G,_H],
    domain(Vars,1,10000),
    sum(Vars, #=, Sum),

    %% a)
    LO = [down], % a opcao mais eficiente para este problema em particular

    % resposta tambem aceite para cotacao maxima apesar de menos eficiente:
    % LO = [maximize(Sum), down],

    % down faz com que o labeling comece por atribuir o valor mais alto do dominio,
resultando numa solucao optima para este problema apenas com um labeling

    labeling(LO, Vars),
    fd_statistics, write(sum-Sum), nl.


p7b(Vars) :-
    Vars = [A,_B,C,D,E,_F,_G,H],
    domain(Vars,1,10000),
    sum(Vars, #=, Sum),

    D #= C - E,
    D #< A,
    D #> H,

    %% b)
    LO = [maximize(Sum), down, ffc], % resposta aceite para cotacao maxima

    % LO = [down,ffc],
    % a mais eficiente, mas aqui nao era obvio que o labeling encontra a resposta optima 'a
primeira tentativa
```

```
    % ffc e' superior a ff porque a variavel D tem mais restrictoes que as outras todas e,
como tal, e' a que mais condiciona a solucao. Este maior numero de restricoes pode nao se
reflectir num dominio mais pequeno em comparacao com as outras variaveis, dai que ffc
garanta que em caso de empate a variavel com mais restricoes (D) e' explorada primeiro.

    labeling(LO, Vars),
    fd_statistics, write(sum-Sum), nl.


% ---------------------------------------------


% modelar as pecas como "tarefas" e usar o cumulative (explorado nas aulas teoricas) permite
tirar partido desta restricao especializada (bastante eficiente) e resolver o problema em
poucas linhas

p8(Ss) :-
    Ss = [S1,S2,S3,S4,S5,S6],
    Es = [E1,E2,E3,E4,E5,E6],

    domain(Ss, 0, 24),
    domain(Es, 1, 25), % a)  % nenhuma "tarefa" pode terminar depois de 25

    Tasks = [   task(S1, 2, E1, 1, 0),
                task(S2, 3, E2, 1, 0),
                task(S3, 5, E3, 1, 0),
                task(S4, 1, E4, 1, 0),
                task(S5, 3, E5, 1, 0),
                task(S6, 6, E6, 1, 0)],

    % c)
    S5 #> S1,

    % d)
    S3 #> S1  #=>  S3 #= 4,

    % b)
    abs(S1-E6) #> 5,
    abs(S6-E1) #> 5,

    cumulative( Tasks ),

    labeling([], Ss).


% ---------------------------------------------


p9(Seq, TotalCost, N) :-
    length(Seq, N),
    domain(Seq, 1, 4),

    % group_vars agrupa pares consecutivos da sequencia e associa a cada par um custo
formando um tripleto;
    % os tripletos serao usados pelo table para indicar transacoes validas e o seu custo
    group_vars(Seq, Costs, Grp),
```

```
    sum(Costs, #=, TotalCost),

    % validacao das precedencias e calculo do custo
    % de forma simples e eficiente utilizando o predicado table
    table(Grp,
            [[1,2,5],[1,3,5],[1,4,7],
             [2,1,3],[2,4,4],
             [3,1,2],[3,4,6],
             [4,1,9],[4,2,5]]),

    %% alternativa: usando o case
    % case([A,B,C], Grp,
            % [node(0,A,[(1..1)-1, (2..2)-2, (3..3)-3, (4..4)-4]),
             % node(1,B,[(2..3)-5, (4..4)-6]),
             % node(2,B,[(1..1)-7, (4..4)-8]),
             % node(3,B,[(1..1)-9, (4..4)-10]),
             % node(4,B,[(1..1)-11, (2..2)-12]),
             % node(5,C,[(5..5)]),
             % node(6,C,[(7..7)]),
             % node(7,C,[(3..3)]),
             % node(8,C,[(4..4)]),
             % node(9,C,[(2..2)]),
             % node(10,C,[(6..6)]),
             % node(11,C,[(9..9)]),
             % node(12,C,[(5..5)])]),

    % as restricoes anteriores podiam alternativamente ser substituidas por um automaton


    % quantidades por tipo de peca (global_cardinality mais eficiente que varios counts)
    global_cardinality(Seq, [1-Um, 2-Dois, 3-Tres, 4-Quatro]),

    % restricoes de limites nas quantidades (operador in mais eficiente que #< e #>)
    Um in 5..15,
    Dois in 2..6,
    Tres in 5..10,
    Quatro in 7..12,

    % b)
    % Automato que conta o numero de sequencias 4-2 depois de aparecer uma 3-4-2
    % uma alternativa igualmente valida (e provavelmente mais eficiente mas um pouco mais
trabalhosa) seria um automato que forcasse a relacao de forma estatica
    automaton(Seq, _, Seq,
        [source(s), sink(a), sink(b), sink(s2), sink(a2)],
        [arc(s,1,s), arc(s,2,s), arc(s,3,a), arc(s,4,s),
         arc(a,1,s), arc(a,2,s), arc(a,4,b),
         arc(b,1,s), arc(b,3,s), arc(b,2,s2),
         arc(s2,1,s2), arc(s2,2,s2), arc(s2,3,s2), arc(s2,4,a2),
         arc(a2,1,s2), arc(a2,2,s2,[C+1]), arc(a2,3,s2), arc(a2,4,s2)
         ], [C],[0],[NBad]),
    NBad #=< 1, % numero de 4-2 a seguir a 3-4-2 têm de ser inferior ou igual a um
    %% fim da b) %%

    labeling([minimize(TotalCost)], Seq).
```

```
% group_vars(Seq, Costs, Out)
group_vars([_], [], []).
group_vars([A,B|RSeq], [C|RCost], [[A,B,C]|ROut]) :-
    group_vars([B|RSeq], RCost, ROut).
```