FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

EXERCÍCIOS DE PROGRAMAÇÃO EM LÓGICA COM RESTRIÇÕES

LUÍS PAULO REIS

LICENCIATURA EM ENGENHARIA INFORMÁTICA E COMPUTAÇÃO

PROGRAMAÇÃO EM LÓGICA - 3º ANO

NOVEMBRO DE 2004

Introdução à Programação em Lógica Com Restrições

IPLR 1. Problema do Quadrado Mágico NxN

O Problema do quadrado mágico consiste em preencher um quadrado com NxN casa, com os números entre 1 e NxN (cada número utilizado uma única vez) de forma a que a soma das linhas, colunas e diagonais (principais) sejam idênticas.

a) Resolva as versões 3x3 e 4x4 do problema

b) Generalize para NxN

Solução 1 a):

```
:-use_module(library(clpfd)).

magic(Vars):-
    Vars=[A1,A2,A3,A4,A5,A6,A7,A8,A9],
    domain(Vars,1,9),
    %Soma is (9+1)*3//2,   % Aumenta a Eficiência
    all_distinct(Vars),
    A1+A2+A3 #= Soma,
    A4+A5+A6 #= Soma,
    A7+A8+A9 #= Soma,
    A1+A4+A7 #= Soma,
    A2+A5+A8 #= Soma,
    A3+A6+A9 #= Soma,
    A1+A5+A9 #= Soma,
    A3+A5+A7 #= Soma,
    % A1 #< A2, A1 #< A3, A1 #< A4, A2 #< A4,  % Eliminar simetrias
    labeling([],Vars).
```

IPLR 2. O "Zebra Puzzle"

Este é um puzzle tradicional da programação em lógica. Há cinco casas com cinco cores diferentes. Em cada casa, vive uma pessoa de nacionalidade diferente, tendo uma bebida, uma marca de cigarros e um animal favoritos. A configuração é:

- O Inglês vive na casa vermelha
- O Espanhol tem um cão
- O Norueguês vive na primeira casa a contar da esquerda
- Na casa amarela, o dono gosta de Marlboro
- O homem que fuma Chesterfields vive na casa ao lado do homem que tem uma raposa
- O Norueguês vive ao lado da casa Azul

- O homem que fuma Winston tem uma iguana
- O fumador de Luky Strike bebe sumo de laranja
- O Ucraniano bebe chá
- O Português fuma SG Lights
- Fuma-se Marlboro na casa ao lado da casa onde há um cavalo
- Na casa verde, a bebida preferida é o café
- A casa verde é imediatamente à direita (à sua direita) da casa branca
- Bebe-se leite na casa do meio

A pergunta é: Onde vive a Zebra, e em que casa se bebe água?

Construir um programa CLP que permita resolver o "Zebra Puzzle".

Solução:

```
:- use_module(library(clpfd)).  % Biblioteca CLP(FD) do SICStus

zebra(Zeb,Agu):-
% Definição das Variáveis e domínios
      Sol = [Nac,Ani,Beb,Cor,Tab],
      Nac = [Ing, Esp, Nor, Ucr, Por],
      Ani = [Cao, Rap, Igu, Cav, Zeb],
      Beb = [Sum, Cha, Caf, Lei, Agu],
      Cor = [Verm,Verd,Bran,Amar,Azul],
      Tab = [Che, Win, LS, SG, Mar],
      %flatten(Sol,List),
      List=[Ing, Esp, Nor, Ucr, Por, Cao, Rap, Igu, Cav, Zeb, Sum, Cha, Caf,
            Lei, Agu, Verm,Verd,Bran,Amar,Azul,Che, Win, LS, SG, Mar],
      domain(List,1,5),

% Colocacao das Restrições
      all_different(Nac),
      all_different(Ani),
      all_different(Beb),
      all_different(Cor),
      all_different(Tab),
      Ing #= Verm,
      Esp #= Cao,
      Nor #= 1,
      Amar #= Mar,
      abs(Che-Rap) #= 1,    % Che #= Rap+1 #\/ Che #= Rap-1
      abs(Nor-Azul) #= 1,
      Win #= Igu,
      LS #= Sum,
      Ucr #= Cha,
      Por #= SG,
      abs(Mar-Cav) #= 1,
      Verd #= Caf,
      Verd #= Bran+1,
      Lei #= 3,

% Pesquisa da solução
      labeling([],List),
      write(Sol),nl.
```

IPLR 3. Problema das N-Rainhas NxN

Construa um programa CLP que permita resolver o problema das N-Rainhas. Este problema consiste em colocar, num tabuleiro com NxN casa, N rainhas (de xadrez), sem que nenhuma rainha ataque uma outra rainha posicionada no tabuleiro (isto é, na horizontal, vertical ou diagonal).

a) Resolva a versão 4x4 do problema

b) Generalize para NxN

Solução 3 a):

```
:-use_module(library(clpfd)).  % Não esquecer de colocar em todos os programas!

nqueens(Cols):-
    Cols=[A1,A2,A3,A4],
    domain(Cols,1,4),
    all_distinct(Cols),              % A1#\=A2, A1#\A3, A1#\A4, A2#\A3, A2#\A4, A3#\A4,
    A1#\=A2+1, A1#\=A2-1, A1#\=A3+2, A1#\=A3-2, A1#\=A4+3, A1#\=A4-3,
    A2#\=A3+1, A2#\=A3-1, A2#\=A4+2, A2#\=A4-2,
    A3#\=A4+1, A3#\=A4-1,
    labeling([],Cols).
```

Solução 3 b):

```
:-use_module(library(clpfd)).

nqueens(N,Cols):-
    length(Cols,N),
    domain(Cols,1,N),
    constrain(Cols),
  % all_distinct(Cols),   % Redundante mas diminui o tempo de resolução
    labeling([],Cols).

constrain([]).
constrain([H | RCols]):-
    safe(H,RCols,1),
    constrain(RCols).

safe(_,[],_).
safe(X,[Y | T], K) :-
    noattack(X,Y,K),
    K1 is K + 1,
    safe(X,T,K1).

noattack(X,Y,K) :-
    X #\= Y,
    X + K #\= Y,
    X - K #\= Y.

% Test:  nqueens(4,C).
% C = [2, 4, 1, 3]     More? (;)
% C = [3, 1, 4, 2]     More? (;)
% no (more) solution.
```

IPLR 4 Problema dos Criptogramas

O Problema dos *CRIPTOGRAMAS* consiste em atribuir dígitos decimais às letras, de modo a que a respectiva soma seja válida. Construa um programa CLP para resolver os seguintes criptogramas:

a) `puzzle(3,[0,S,E,N,D],[0,M,O,R,E],[M,O,N,E,Y]).`

b) `puzzle(1,[D,O,N,A,L,D],[G,E,R,A,L,D],[R,O,B,E,R,T]).`

c) `puzzle(2,[0,C,R,O,S,S],[0,R,O,A,D,S],[D,A,N,G,E,R]).`

d) Construa um programa CLP para resolver criptogramas genéricos.

Solução 4 a): Solução Simples

```
:-use_module(library(clpfd)).
send(Vars):-
    Vars=[S,E,N,D,M,O,R,Y],
    domain(Vars,0,9),
    all_different(Vars),
    S #\= 0, M #\= 0,
    S*1000 + E*100 + N*10 + D + M*1000 + O*100 + R*10 + E #=
          M*10000 + O*1000 + N*100 + E*10 + Y,
    labeling([],Vars).
```

Solução 4 a): Mais Eficiente

```
:-use_module(library(clpfd)).
send(Vars):-
    Vars=[S,E,N,D,M,O,R,Y],
    domain(Vars,0,9),
    domain([C1,C2,C3,C4],0,1),
    all_distinct(Vars),
    S #\= 0, M #\= 0,
    D + E #= Y+ C1*10,
    N + R + C1 #= E+ C2*10,
    E + O + C2 #= N+ C3*10,
    S + M + C3 #= O+ C4*10,
    C4 #= M,
    labeling([ff],Vars).
```

IPLR 5 Guardas no Forte

Doze guardas são colocados a guardar um forte com quatro salas em cada lado. Se um guarda estiver numa sala do lado só pode ver esse lado, enquanto se estiver num canto pode ver dois lados. O objectivo é construir um programa em PLR capaz de colocar 12 guardas no forte de forma a que cinco guardas vigiem cada lado.

IPLR 6. Soma e Produto

Que conjuntos de três números têm a sua soma igual ao seu produto?

Solução:

```
sol(A,B,C):-
    domain([A,B,C],1,1000),
    A*B*C #= A+B+C,
    % C#>=B, B#>=A,   %Eliminar simetrias
    labeling([],[A,B,C]).
```

IPLR 7. Peru Assado

Uma factura antiga revela que setenta e dois perus foram comprados por "-67-" Escudos. O primeiro e o último algarismo estão ilegíveis. Construa um programa PLR capaz de determinar quanto é que, na época, custava cada peru.

IPLR 8. O Puto na Mercearia

Um rapaz vai à Mercearia comprar um pacote de arroz, um saco de batatas, um pacote de esparguete e uma lata de Atum. O merceeiro (que por sinal era muito careiro) diz-lhe que são 7.11€. O rapaz paga e quando já vai a sair, o merceeiro diz-lhe: "Espera! Enganei-me e multipliquei o valor dos produtos em vez de somar. Mas afinal somando-os dá na mesma 7.11€!". Sabendo que o preço de dois dos produtos eram múltiplos de 10 cêntimos e que as batatas eram mais caras do que o atum e este mais caro do que o arroz e que o produto mais barato era o esparguete, qual era o preço de cada um dos produtos?

IPLR 9. Zero Zeros

Como é possível escrever 1 000 000 000 como um produto de dois factores, cada um dos quais não contendo qualquer zero?