Inteligência Artificial

Trabalho 2

Resolução de problemas de satisfação de restrições

Discentes:

João Santos nº 29634

Évora 2016

Introdução.

Neste trabalho foi-me pedido que encarasse o problema do quadrado mágico como um problema de CSP. Neste quadrado, todos os números são diferentes e a soma das linhas, colunas e diagonais principais é igual. (ficheiros quadradomagico.pl e back.pl)

Também foi pedido a resolução do sudoku em que em todas as linhas e colunas os números são diferentes (de 1 a 9) e também nos 9 quadrados 3x3 que constituem o tabuleiro. (ficheiros sudoku.pl e backsudoku.pl)

1 – Respostas.

a).

Para encarar o problema como satisfação de restrições vai ser necessário definir o estado inicial, as coordenadas possíveis e o domínio.

Os estados são definidos com um tuplo do tipo v(coordenada, domínio, valor).

O estado inicial é definido com o quadrado mágico vazio, com a primeira lista por afectar e a segunda lista com as posições afectadas (vazia inicialmente).

```
estado_inicial(e([
v(c(1,1), [1, 2, 3, 4, 5, 6, 7, 8, 9], _),
v(c(1,2), [1, 2, 3, 4, 5, 6, 7, 8, 9], _),
v(c(1,3), [1, 2, 3, 4, 5, 6, 7, 8, 9], _),
v(c(2,1), [1, 2, 3, 4, 5, 6, 7, 8, 9], _),
v(c(2,2), [1, 2, 3, 4, 5, 6, 7, 8, 9], _),
v(c(2,3), [1, 2, 3, 4, 5, 6, 7, 8, 9], _),
v(c(3,1), [1, 2, 3, 4, 5, 6, 7, 8, 9], _),
v(c(3,2), [1, 2, 3, 4, 5, 6, 7, 8, 9], _),
v(c(3,3), [1, 2, 3, 4, 5, 6, 7, 8, 9], _)], [])).
```

Para as restrições primeiro verificou-se se os números eram diferentes dentro do quadrado e de seguida verificar se estavam preenchidos e se a soma efectivamente era igual.

```
ve_restricoes(e(Nafect,Afect)):-

  \+ (member(v(c(I,J),_,Vj), Afect),

      member(v(c(A,B),_,Vk), Afect),

      A \= I,

      J \= B,

      Vk = Vj),

  (\+ preenchidos(Nafect) ; soma(Afect)).


soma(Afect):-

      member(v(c(1,1),_,V11), Afect),

      member(v(c(1,2),_,V12), Afect),

      member(v(c(1,3),_,V13), Afect),

      SomaL1 is V11+V12+V13,

      member(v(c(2,1),_,V21), Afect),

      member(v(c(2,2),_,V22), Afect),

      member(v(c(2,3),_,V23), Afect),

      SomaL2 is V21+V22+V23,

      member(v(c(3,1),_,V31), Afect),

      member(v(c(3,2),_,V32), Afect),

      member(v(c(3,3),_,V33), Afect),

      SomaL3 is V31+V32+V33,


      member(v(c(1,1),_,V11), Afect),

      member(v(c(2,1),_,V21), Afect),

      member(v(c(3,1),_,V31), Afect),

      SomaC1 is V11+V21+V31,
```

```
        member(v(c(1,2),_,V12), Afect),
        member(v(c(2,2),_,V22), Afect),
        member(v(c(3,2),_,V32), Afect),
        SomaC2 is V12+V22+V32,
        member(v(c(1,3),_,V13), Afect),
        member(v(c(2,3),_,V23), Afect),
        member(v(c(3,3),_,V33), Afect),
        SomaC3 is V13+V23+V33,

        member(v(c(1,1),_,V11), Afect),
        member(v(c(2,2),_,V22), Afect),
        member(v(c(3,3),_,V33), Afect),
        SomaD1 is V11+V22+V33,
        member(v(c(1,3),_,V13), Afect),
        member(v(c(2,2),_,V22), Afect),
        member(v(c(3,1),_,V31), Afect),
        SomaD2 is V13+V22+V31,

        SomaL1 = SomaL2,
        SomaL2 = SomaL3,
        SomaL3 = SomaC1,
        SomaC1 = SomaC2,
        SomaC2 = SomaC3,
        SomaC3 = SomaD1,
        SomaD1 = SomaD2.

Preenchidos([]).
```

Para a resolução em backtracking, é necessária a afectação de valores do dominio às diferentes posições e a sua confirmação através das restrições definidas. Para esse fim serve o seguinte código:

```
p(Prg):- consult(Prg),estado_inicial(E0),back(E0,A), esc(A).

back(e([],A),A).

back(E,Sol):- sucessor(E,E1), ve_restricoes(E1),

back(E1,Sol).

sucessor(e([v(N,D,V)|R],E),e(R1,[v(N,D,V)|E])):- member(V,D).
```

c).

Para a resolução em forward checking é necessário, ao fazer uma atribuição, remover dos dominios com a qual interfere, todas as opções que causam conflito. Neste caso, ao atribuir um valor a uma posição, é necessário remover do dominio de toda a linha toda a coluna e todo o quadrado associado a essa posição esse mesmo valor. No predicado sucessor foi adicionado o “remove”.

```
sucessor(e([v(N,D,V)|R],E),e(R1,[v(N,D,V)|E])):- member(V,D),
remove(N,V,R,R1).

remove(c(I,J),V,R,R3):-

  linha(I,V,R,R1),

  coluna(J,V,R1,R2),

  diagonais(c(I,J),V,R2,R3).
```

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

linha(Linha,Valor,[v(c(Linha,Col),D,_)|R],[v(c(Linha,Col),D2,_)|
R1]):-

  removeLista(Valor,D,D2),

  linha(Linha,Valor,R,R1),!.


linha(Linha,Valor,[I|R],[I|R1]):-

  linha(Linha,Valor,R,R1),!.


coluna(_,_,[],[]).

coluna(Col,Valor,[v(c(Linha,Col),D,_)|R],[v(c(Linha,Col),D2,_)|
R1]):-

  removeLista(Valor,D,D2),

  coluna(Col,Valor,R,R1),!.


coluna(Col,Valor,[I|R],[I|R1]):-

  coluna(Col,Valor,R,R1),!.


diagonais(N,V,R,R1):-

  diagonal(N,Lista),

  removeDiagonais(Lista,V,R,R1).


removeDiagonais([],_,R,R).


removeDiagonais([N|L],V,R,R2):-

  member(v(N,_,_),R),

  findRemove(v(N,_,_),V,R,R1),

  removeDiagonais(L,V,R1,R2).
```

```
removeDiagonais([N|L],V,R,R1):-
  \+ member(v(N,_,_),R),
  removeDiagonais(L,V,R,R1).

findRemove(v(N,_,_),V,[v(N,D,L)|R],[v(N,D2,L)|R]):-
  removeLista(V,D,D2),!.

findRemove(v(N,_,_),V,[I|R],[I|R1]):-
  findRemove(v(N,_,_),V,R,R1).

removeLista(_,[],[]):-!.
removeLista(Valor,[Valor|R],R):-!.
removeLista(Valor,[X|R],[X|R1]):-
  removeLista(Valor,R,R1).
```

e).

Quadrado vazio:

```
?- p('quadmagico.pl').

2 7 6
9 5 1
4 3 8
true
```

Posição (2,3) preenchida com um 4:

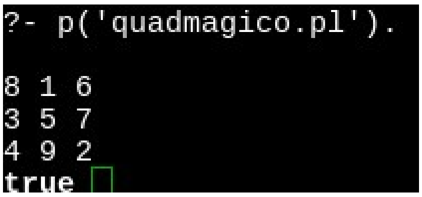
```
?- p('quadmagico.pl').
false.
```

Posição (1,3) preenchida com um 4:

```
?- p('quadmagico.pl').

2 9 4
7 5 3
6 1 8
true .
```

Posição (1,1) preenchida com um 8:



```
?- p('quadmagico.pl').

8 1 6
3 5 7
4 9 2
true
```

2 – Respostas.

a).

Para encarar o problema como satisfação de restrições vai ser necessário definir o estado inicial, as coordenadas possíveis e o domínio.

Os estados são definidos com um tuplo do tipo v(coordenada, domínio, valor).

O estado inicial é definido com o sudoku com algumas posições preenchidas (porque vazio fica em loop infinito à procura de soluções), com a primeira lista com as posições que faltam afectar e a segunda lista com as posições afectadas.

```
estado_inicial(e([


v(c(5,6),[1,2,3,4,5,6,7,8,9],_),

v(c(2,5),[1,2,3,4,5,6,7,8,9],_),

v(c(3,5),[1,2,3,4,5,6,7,8,9],_),

v(c(3,7),[1,2,3,4,5,6,7,8,9],_),

v(c(3,8),[1,2,3,4,5,6,7,8,9],_),

v(c(3,9),[1,2,3,4,5,6,7,8,9],_),

v(c(4,4),[1,2,3,4,5,6,7,8,9],_),

v(c(4,5),[1,2,3,4,5,6,7,8,9],_),

v(c(4,6),[1,2,3,4,5,6,7,8,9],_),

v(c(4,7),[1,2,3,4,5,6,7,8,9],_),

v(c(4,8),[1,2,3,4,5,6,7,8,9],_),
```

```
v(c(4,9),[1,2,3,4,5,6,7,8,9],_),
v(c(5,7),[1,2,3,4,5,6,7,8,9],_),
v(c(5,8),[1,2,3,4,5,6,7,8,9],_),
v(c(5,9),[1,2,3,4,5,6,7,8,9],_),
v(c(9,2),[1,2,3,4,5,6,7,8,9],_)

],

[
v(c(1,1),[1,2,3,4,5,6,7,8,9],5),
v(c(1,2),[1,2,3,4,5,6,7,8,9],8),
v(c(1,3),[1,2,3,4,5,6,7,8,9],6),
v(c(1,4),[1,2,3,4,5,6,7,8,9],3),
v(c(1,5),[1,2,3,4,5,6,7,8,9],7),
v(c(1,6),[1,2,3,4,5,6,7,8,9],4),
v(c(1,7),[1,2,3,4,5,6,7,8,9],9),
v(c(1,8),[1,2,3,4,5,6,7,8,9],1),
v(c(1,9),[1,2,3,4,5,6,7,8,9],2),
v(c(2,1),[1,2,3,4,5,6,7,8,9],1),
v(c(2,2),[1,2,3,4,5,6,7,8,9],3),
…
v(c(9,6),[1,2,3,4,5,6,7,8,9],5),
v(c(9,7),[1,2,3,4,5,6,7,8,9],4),
v(c(9,8),[1,2,3,4,5,6,7,8,9],2),
v(c(9,9),[1,2,3,4,5,6,7,8,9],7)
])).
```

Foram seguidas as restrições descritas na introdução.

```
ve_restricoes(e(_,Afect)):-

  \+ (member(v(c(I,J),_,Vj), Afect),
member(v(c(I,K),_,Vk),Afect),  K \=J,Vk=Vj),

  \+ (member(v(c(I,J),_,Vi), Afect),
member(v(c(K,J),_,Vk),Afect),  K \=I,Vi=Vk),

  ve_quadrado(1, 1, Afect).


ve_quadrado(10, 10, _).

ve_quadrado(I, 10, Afect):-

  I < 8,

  I2 is I+3,

  ve_quadrado(I2,1,Afect).



ve_quadrado(I,J,Afect):-

    I1 is I+1,

    J1 is J+1,

    \+ (member(v(c(I,J),_,Vi), Afect),
member(v(c(I1,J1),_,Vj),Afect),  Vi=Vj),

    I2 is I1+1,

    J2 is J1+1,

    \+ (member(v(c(I1,J1),_,Vi), Afect),
member(v(c(I2,J2),_,Vj),Afect),  Vi=Vj),

    \+ (member(v(c(I1,J1),_,Vi), Afect),
member(v(c(I1,J2),_,Vj),Afect),  Vi=Vj),

    \+ (member(v(c(I1,J1),_,Vi), Afect),
member(v(c(I2,J1),_,Vj),Afect),  Vi=Vj),

    J3 is J2+1,

    ve_quadrado(I, J3, Afect).
```