Trabalho 2 Inteligência Artificial

Sudoku, CSP

- **Estados**

Cada estado é representado por duas listas onde a primeira é a lista de variáveis não instanciadas enquanto a segunda é a de variáveis instanciadas.

- **Variáveis**

As variáveis apresentam a seguinte estrutura, v(c(I), D, V), onde I é a casa a que pertence (entre 1 e 81, inclusive), D o domínio da variável (0..9), e V o valor atribuído.

As variáveis já definidas, têm o seu valor, V, logo atribuído no estado inicial.

João Pedro Castanheira Marcelino Verdilheiro | 38156

Estado inicial

cod.pl

```
estado_inicial(e([
                    v(c(1),[1,2,3,4,5,6,7,8,9],_),
                    v(c(2),[1],1),
                    v(c(3),[1,2,3,4,5,6,7,8,9],_),
                    v(c(4),[1,2,3,4,5,6,7,8,9],_),
                    v(c(5),[1,2,3,4,5,6,7,8,9],_),
                    v(c(6),[8],8),
                    v(c(7),[1,2,3,4,5,6,7,8,9],_),
                    v(c(8),[1,2,3,4,5,6,7,8,9],_),
                    v(c(9),[3],3),
                    v(c(10),[1,2,3,4,5,6,7,8,9],_),
                    v(c(11),[1,2,3,4,5,6,7,8,9],_),
                    v(c(12),[1,2,3,4,5,6,7,8,9],_),
                    v(c(13),[6],6),
                    v(c(14),[1,2,3,4,5,6,7,8,9],_),
                    v(c(15),[9],9),
                    v(c(16),[1,2,3,4,5,6,7,8,9],_),
                    v(c(17),[1,2,3,4,5,6,7,8,9],_),
                    v(c(18),[1,2,3,4,5,6,7,8,9],_),
                    v(c(19),[7],7),
                    v(c(20),[1,2,3,4,5,6,7,8,9],_),
                    v(c(21),[1,2,3,4,5,6,7,8,9],_),
                    v(c(22),[1,2,3,4,5,6,7,8,9],_),
                    v(c(23),[1,2,3,4,5,6,7,8,9],_),
                    v(c(24),[1,2,3,4,5,6,7,8,9],_),
                    v(c(25),[1,2,3,4,5,6,7,8,9],_),
                    v(c(26),[1,2,3,4,5,6,7,8,9],_),
                    v(c(27),[4],4),
                    v(c(28),[1,2,3,4,5,6,7,8,9],_),
```

- **Restrições**

As restrições são tratadas com três focos especiais, sendo que para a primeira, verifica que um número não se repete por coluna. A segunda verifica o mesmo, mas por linha e por fim a terceira verifica que por cada quadrado de 3x3 do sudoku não existem números repetidos.

```
ve_restricoes(E):- (ve_restricoes1(E) , ve_restricoes2(E), ve_restricoes3(E)).

% Não se repete por coluna
ve_restricoes1(e(Nafec,Afect)):- \+ (member(v(c(I),Di,Vi), Afect), member(v(c(J),Dj,Vj),Afect),  I \=J,
                                    % mod 9 verifica a que coluna pertence
                                    Vi=Vj , I1 is mod(I,9), J1 is mod(J,9), I1=J1 ).

ve_restricoes2(e(Nafec,Afect)):- \+ (member(v(c(I),Di,Vi), Afect), member(v(c(J),Dj,Vj),Afect),  I \=J,
                                    Vi=Vj ,
                                    % no caso de ser o ultimo da linha e ser divisor o resto sera
                                    divisor(I, I1), divisor(J, J1),
                                    I2 is I1 // 9, J2 is J1 // 9, J2 = I2).

ve_restricoes3(e(Nafec,Afect)):- \+ (member(v(c(I),Di,Vi), Afect), member(v(c(J),Dj,Vj),Afect),  I \=J,
                                       Vi = Vj, casas(I,J) ).


%ve_restricoes(e(Nafect,[A])).

% Altera o numero se o mod for 0 (se este estiver na ultima linha),
% evitando a ultima coluna onde I//9 seria mais um que a restante linha
divisor(I, I1):- N is I mod 9, altera(N,I, N1), I1 is N1.
altera(N, I, N1):- N = 0, N1 is I - 1.
```

```
casas(I,J):-member(I,[1,2,3,10,11,12,19,20,21]), member(J,[1,2,3,10,11,12,19,20,21]).
casas(I,J):-member(I,[4,5,6,13,14,15,22,23,24]), member(J,[4,5,6,13,14,15,22,23,24]).
casas(I,J):-member(I,[7,8,9,16,17,18,25,26,27]), member(J,[7,8,9,16,17,18,25,26,27]).

casas(I,J):-member(I,[28,29,30,37,38,39,46,47,48]), member(J,[28,29,30,37,38,39,46,47,48]).
casas(I,J):-member(I,[31,32,33,40,41,42,49,50,51]), member(J,[31,32,33,40,41,42,49,50,51]).
casas(I,J):-member(I,[34,35,36,43,44,45,52,53,54]), member(J,[34,35,36,43,44,45,52,53,54]).

casas(I,J):-member(I,[55,56,57,64,65,66,73,74,75]), member(J,[55,56,57,64,65,66,73,74,75]).
casas(I,J):-member(I,[58,59,60,67,68,69,76,77,78]), member(J,[58,59,60,67,68,69,76,77,78]).
casas(I,J):-member(I,[61,62,63,70,71,72,79,80,81]), member(J,[61,62,63,70,71,72,79,80,81]).
```

- **Operador Sucessor**

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

- **Resolução**

Para a resolução do Sudoku apresentado foi utilizada uma versão ligeiramente mais simplificada para que fosse possível diminuir o tempo de processamento, sendo ainda assim uma versão bastante semelhante á apresentada no enunciado.

Para correr o programa basta o seguinte comando, “p.” no ficheiro “cod.pl”.

Proposta de problema:

	1				8			3
			6		9			
7								4
					4			
				3	6		1	8
8			9					
			7					
							3	

Solução:

```
214578693
358649127
769123584
123854769
597236418
846917352
631782945
972465831
485391276
```

Na resolução deste problema foi utilizado o algoritmo backtracking e implementado uma modificação para que este faça forward checking embora esta implementação não fosse a ideal.

Foi também utilizada a mesma versão do algoritmo inicial para o resultado final sendo que não foi possível melhorar a sua complexidade como pedido no final do enunciado (d).