

Resolução de problemas como problemas satisfação de restrições

João Matos (32409), João Rouxinol (44451) e André Rato (45517)

14 de maio de 2022

1 Sudoku como um CSP

1.1 O problema

1.1.1 Estados

Para a representação do problema, os estados seguem todos a estrutura ”e(LNI, LI)”, em que LNI é a lista de variáveis não instanciadas e LI é a lista de variáveis instanciadas.

1.1.2 Variáveis

As variáveis apresentam a estrutura ”v(p(X, Y), D, V)”, onde X e Y são as coordenadas da variável, D o domínio e V o valor.

1.1.3 Restrições

O predicado responsável por verificar as restrições é o predicado `ve_restricoes`, que verifica as linhas, as colunas e os quadrantes, de modo a que todos os elementos destes três grupos sejam diferentes.

1.1.4 Estado inicial

O estado inicial segue a estrutura dos estados do problema, em que o primeiro elemento do predicado é composto por uma lista com todas as variáveis vazias, e o segundo é composto por uma lista com todas as variáveis já preenchidas (as já instanciadas).

```
%% e = estado
e = e([v(1,1,''), v(1,2,''), v(1,3,''), v(1,4,''), v(1,5,''), v(1,6,''), v(1,7,''), v(1,8,''), v(1,9,''), v(2,1,''), v(2,2,''), v(2,3,''), v(2,4,''), v(2,5,''), v(2,6,''), v(2,7,''), v(2,8,''), v(2,9,''), v(3,1,''), v(3,2,''), v(3,3,''), v(3,4,''), v(3,5,''), v(3,6,''), v(3,7,''), v(3,8,''), v(3,9,''), v(4,1,''), v(4,2,''), v(4,3,''), v(4,4,''), v(4,5,''), v(4,6,''), v(4,7,''), v(4,8,''), v(4,9,''), v(5,1,''), v(5,2,''), v(5,3,''), v(5,4,''), v(5,5,''), v(5,6,''), v(5,7,''), v(5,8,''), v(5,9,''), v(6,1,''), v(6,2,''), v(6,3,''), v(6,4,''), v(6,5,''), v(6,6,''), v(6,7,''), v(6,8,''), v(6,9,''), v(7,1,''), v(7,2,''), v(7,3,''), v(7,4,''), v(7,5,''), v(7,6,''), v(7,7,''), v(7,8,''), v(7,9,''), v(8,1,''), v(8,2,''), v(8,3,''), v(8,4,''), v(8,5,''), v(8,6,''), v(8,7,''), v(8,8,''), v(8,9,''), v(9,1,''), v(9,2,''), v(9,3,''), v(9,4,''), v(9,5,''), v(9,6,''), v(9,7,''), v(9,8,''), v(9,9,'')], [v(1,1,''), v(1,2,''), v(1,3,''), v(1,4,''), v(1,5,''), v(1,6,''), v(1,7,''), v(1,8,''), v(1,9,''), v(2,1,''), v(2,2,''), v(2,3,''), v(2,4,''), v(2,5,''), v(2,6,''), v(2,7,''), v(2,8,''), v(2,9,''), v(3,1,''), v(3,2,''), v(3,3,''), v(3,4,''), v(3,5,''), v(3,6,''), v(3,7,''), v(3,8,''), v(3,9,''), v(4,1,''), v(4,2,''), v(4,3,''), v(4,4,''), v(4,5,''), v(4,6,''), v(4,7,''), v(4,8,''), v(4,9,''), v(5,1,''), v(5,2,''), v(5,3,''), v(5,4,''), v(5,5,''), v(5,6,''), v(5,7,''), v(5,8,''), v(5,9,''), v(6,1,''), v(6,2,''), v(6,3,''), v(6,4,''), v(6,5,''), v(6,6,''), v(6,7,''), v(6,8,''), v(6,9,''), v(7,1,''), v(7,2,''), v(7,3,''), v(7,4,''), v(7,5,''), v(7,6,''), v(7,7,''), v(7,8,''), v(7,9,''), v(8,1,''), v(8,2,''), v(8,3,''), v(8,4,''), v(8,5,''), v(8,6,''), v(8,7,''), v(8,8,''), v(8,9,''), v(9,1,''), v(9,2,''), v(9,3,''), v(9,4,''), v(9,5,''), v(9,6,''), v(9,7,''), v(9,8,''), v(9,9,'')]);
```

Figura 1: Estado inicial

1.1.5 Operador sucessor

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

Figura 2: Operador sucessor

1.2 Resolução com algoritmo de backtracking

```
yes
| ?- sudoku_back.
6 . 1 . 9 . 4 . 2 . 8 . 5 . 7 . 3
5 . 3 . 4 . 6 . 7 . 9 . 1 . 8 . 2
7 . 2 . 8 . 3 . 1 . 5 . 9 . 6 . 4
3 . 6 . 2 . 1 . 8 . 4 . 7 . 9 . 5
9 . 7 . 5 . 2 . 3 . 6 . 4 . 1 . 8
8 . 4 . 1 . 9 . 5 . 7 . 3 . 2 . 6
4 . 9 . 3 . 7 . 6 . 2 . 8 . 5 . 1
2 . 5 . 7 . 8 . 4 . 1 . 6 . 3 . 9
1 . 8 . 6 . 5 . 9 . 3 . 2 . 4 . 7
```

Figura 3: Estado final do tabuleiro após utilização do algoritmo de backtracking

1.3 Resolução com algoritmo de forward checking

```
| ?-
sudoku_forward.
Fatal Error: global stack overflow (size: 32768 Kb, reached: 327
65 Kb, environment variable used: GLOBALSZ)
```

Figura 4: Output após utilização do algoritmo de forward checking

Isto acontece pois é excedida a memória disponibilizada para a execução do PROLOG.