Pentalath in Prolog

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo xx:
Inês Catarina Ferreira Teixeira - up201404592
Pedro Duarte Costa - up201403291

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

16 de Outubro de 2016

1 O Jogo Pentalath

Pentalath é um jogo de tabuleiro de dois jogadores que tem como finalidade isolar o inimigo de forma a capturar as suas peças e formar linhas de cinco peças, ou mais, da mesma cor.

Este jogo, criado pelo computador 'Ludi' em 2007 junta várias regras de jogos de tabuleiro tradicionais como o '5 em Linha' e o 'Capture Go', mas com algumas particularidades como a sua forma, um trapézio com o total de 70 células hexagonais, como é possível ver na figura 1, abaixo representada.

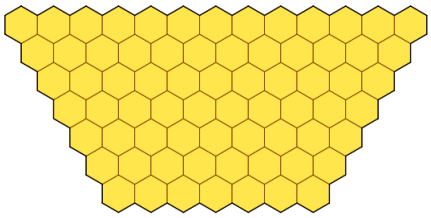

Figura 1: Tabuleiro do jogo

O tabuleiro começa vazio, no entanto, cada jogador tem 40 peças, brancas ou pretas.

Na figura 2,pode-se ver um exemplo de uma jogada onde a peça branca 6 interseta a hipótese de a equipa preta formar uma linha e vencer:

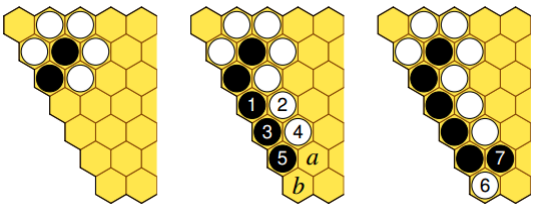

Figura 2: Exemplo de jogada

A cada jogada, se uma peça for rodeada por peças da cor oposta, então a peça é capturada do tabuleiro, como é possível verificar na figura 3. As peças pretas, após a colocação da peça branca com a letra 'f', estão rodeadas pelas peças brancas, dessa forma devem ser retiradas do tabuleiro e não podem ser jogadas.

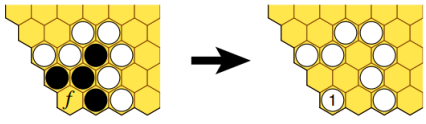

Figura 3: Exemplo de captura de peças inimigas

Desta forma, Pentalath é um jogo de estratégia, onde o jogador deve ter em

mente diferentes objetivos- estender uma linha de cinco peças iguais enquanto captura as peças do inimigo.

2 Representação do Estado do Jogo

O tabuleiro é formado por uma lista de listas, que cada linha corresponde a uma linha do tabuleiro, com uma particularidade, cada linha tem menos uma casa que a anterior de forma a assumir a forma trapezoidal do tabuleiro do Pentolath. Desta forma, o valor predefinido, segundo as regras, é treze casas para a primeira linha e a última apenas sete.

De forma a dinamizar o jogo e a criar vários níveis de dificuldades, o tamanho do tabuleiro é dinâmico, sendo criada a lista de listas correspondente ao tamanho pedido através dos seguintes predicados representados no listing 1:

```
1  init_list(R, [ ]) :-
2      R =< 0, !.
3  init_list(R, [_ | T]) :-
4      R > 0,
5      R2 is R - 1,
6      init_list(R2, T).
7
8  init_matrix(_, R, []) :-
9      R =< 0, !.
10
11 %init_matrix(Columns,Rows,Board)
12 init_matrix(C, R, [H|T]) :-
13     init_list(C,H),
14     R2 is R - 1,
15     init_matrix(C-1, R2, T).
```

Listing 1: Código para criar dinamicamente o tabuleiro

A cada nova linha é introduzida com menos uma casa que a linha anterior.

Cada uma da casas vazias é representada por 'null'.

A matriz inicial, sem peças, para o tabuleiro de tamanho predefinido nas regras é:

board([
[null,null,null,null,null,null,null,null,null,null,null,null,null],
[null,null,null,null,null,null,null,null,null,null,null,null],
[null,null,null,null,null,null,null,null,null,null,null],
[null,null,null,null,null,null,null,null,null,null],
[null,null,null,null,null,null,null,null,null],
[null,null,null,null,null,null,null,null],
[null,null,null,null,null,null,null]
])

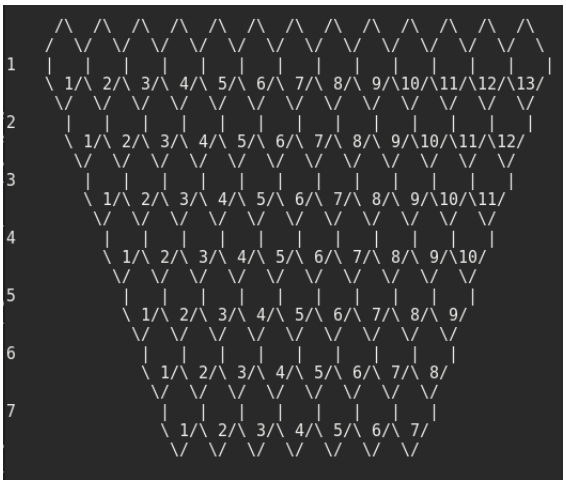

Figura 4: Representação gráfica do tabuleiro vazio.

Exemplo de matriz intermédia, após algumas jogadas:

board([
[null,**black**,null,null,null,null,null,null,null,null,null,null,null],
[null,null,**black**,null,null,null,null,null,null,null,null,null,null],
[null,null,null,null,null,**black**,null,null,null,null,null,null],
[null,null,null,null,null,null,null,null,null,null,null],
[**white**,**white**,**white**,**white**,null,null,null,null,null],
[null,null,null,null,null,null,null,null],
[null,null,null,null,null,null,null]
])

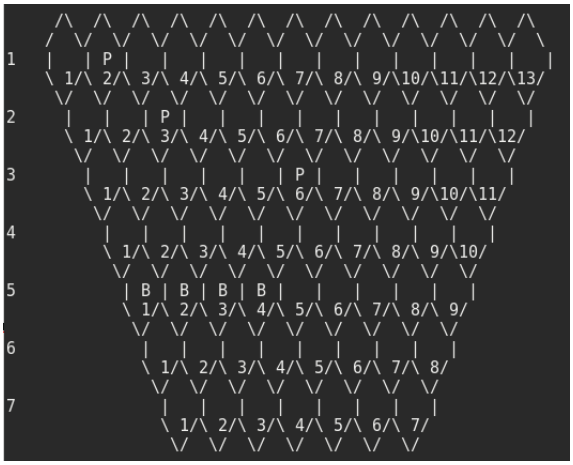

Figura 5: Representação gráfica do tabuleiro vazio.

Exemplo de matriz final, aquando numa situação de vitória pelas peças brancas:

board([
[null,**black**,null,null,null,null,null,null,null,null,null,null,null],
[null,null,**black**,null,**black**,**black**,**black**,null,null,null,null,null,null],
[null,null,**white**,null,null,**black**,null,null,null,null,null,null],
[null,null,null,null,null,null,null,null,null,null,null],
[**white**,**white**,**white**,**white**,**white**,null,null,null,null],
[null,null,null,null,null,null,null,null,null],
[null,null,null,null,null,null,null,null,null]
])

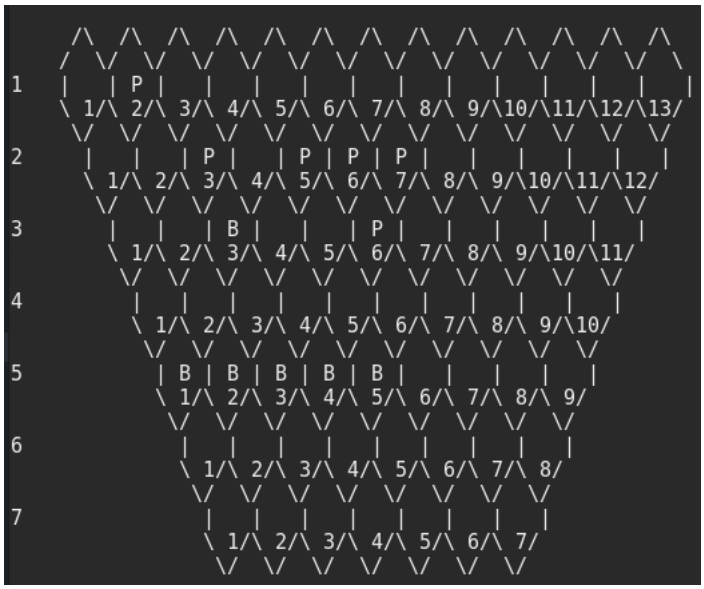

Figura 6: Representação gráfica do tabuleiro vazio.

3 Visualização do Tabuleiro

Para representar os diferentes estados das casas do tabuleiro vão ser usados três símbolos diferentes: a letras 'B' (peças brancas), a letra 'P' (peças pretas) e o ' '(espaço, casa vazia).

A função que imprime o tabuleiro é a display_board, represntada no listing 2, que recebe os seguintes argumentos:

Board,Size+1,InitialIndentation,BooleanForDisplayingTopRow,InitialIndex-1, que se faz acompanhar de outras auxiliares.

```
1 display_walls(S,SI):-
2   S > 0,
3   SI < 9,
4   S1 is S - 1,
5   S2 is SI + 1,
```

```
6   write(''\' '),
7   write(S2),
8   write('/'),
9   display_walls(S1,S2).
10
11  display_walls(S,SI):-
12    S > 0,
13    S1 is S - 1,
14    S2 is SI + 1,
15    write(''\''),
16    write(S2),
17    write('/'),
18    display_walls(S1,S2).
19
20  display_walls(S,SI):-nl.
21
22  display_walls2(S):-
23    S > 0,
24    S1 is S - 1,
25    write(' '\''),
26    write('/ '),
27    display_walls2(S1).
28
29  display_walls2(S):-nl.
30
31  display_spaces(SP):-
32    SP > 0,
33    SP1 is SP - 1,
34    write('  '),
35    display_spaces(SP1).
36
37  display_spaces(SPA):-
38    write('').
39
40  display_board_top(S):-
41    S > 0,
42    S1 is S - 1,
43    write(' /'),
44    write(''\' '),
45    display_board_top(S1).
46
47  display_board_top(S):-nl.
48
49
50  display_board_top2(S):-
51    S > 0,
52    S1 is S - 1,
53    write('/  '),
54    write(''\''),
55    display_board_top2(S1).
56
57  display_board_top2(S):-nl.
58
59
60  display_board([L1|LS],SIZE,SPACE,TOP,SI):-
61    SIZE > 7,
62    SIZE1 is SIZE - 1,
63    SPACE1 is SPACE + 1,
64    TOP > 0,
65    display_spaces(SPACE),
66    display_board_top(SIZE-1),
67    display_spaces(SPACE),
68    display_board_top2(SIZE-1),
69    INDEX is SPACE1 -  2,
70    write(INDEX),
```

```
71    display_spaces(SPACE-1),
72    write(' |'),
73    display_line(L1),nl,
74    display_spaces(SPACE),
75    display_walls(SIZE1,SI),
76    display_spaces(SPACE),
77    display_walls2(SIZE1),
78    display_board(LS,SIZE1,SPACE1,0,SI).
79
80    display_board([L1|LS],SIZE,SPACE,TOP,SI):-
81      SIZE > 7,
82      SIZE1 is SIZE - 1,
83      SPACE1 is SPACE + 1,
84      INDEX is SPACE1 -  2,
85      write(INDEX),
86      display_spaces(SPACE-1),
87      write(' |'),
88      display_line(L1),nl,
89      display_spaces(SPACE),
90      display_walls(SIZE1,SI),
91      display_spaces(SPACE),
92      display_walls2(SIZE1),
93      display_board(LS,SIZE1,SPACE1,0,SI).
94
95  display_board([],SIZE,SPACE1,TOP,SI):-nl.
96
97  display_line([E1|ES]):-
98    traduz(E1,V),
99    write(V),
100   write('|'),
101   display_line(ES).
102
103 display_line([]):-write('').
104
105 traduz(null     ,'   ').
106 traduz(b        ,' B ').
107 traduz(p        ,' P ').
108 traduz(E1       ,'').
```

Listing 2: Code example

Com a seguinte representação gráfica:

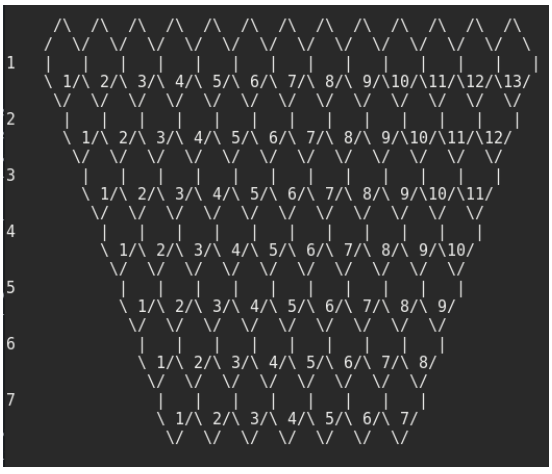

Figura 7: Representação gráfica do tabuleiro

4 Movimentos

O jogador pode colocar uma peça em qualquer casa do tabuleiro, no entanto, após a peça ser colocada, esta não pode ser deslocada nem retirada, salvo o caso de a peça ser capturada por peças de cor inimiga. Estas jogadas serão feitas com as seguintes funções:

- putPiece(Row, Column) - posição no referencial onde a peça será colocada.
- removePiece(Row, Column) - posição no referencial em que uma peça deve ser retirada por ter sido capturada por peças inimigas.