Meerkats

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo Meerkats_2:

Daniel Silva Reis – up201308586
Guilherme Vieira Pinto – up201305803

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

1. Introdução

Como forma de avaliar os nossos conhecimentos da cadeira de Programação em Lógica, foi proposto a realização de um trabalho recorrendo à linguagem Prolog. Das várias propostas de tema, decidimos escolher o Meerkats pois é um jogo com alguma complexidade e estratégia que possibilita a implementação de inteligência artificial como é pretendido pela cadeira.

2. O jogo Meerkats

2.1 História

Meerkats é um jogo de estratégia baseado no comportamento de bandos de suricatas. Tendo em conta que esta espécie de mamífero é caraterizada por sobreviver em comunidades organizadas, com cada elemento desempenhando um papel indispensável para a segurança do bando, permite que seja elaborado um jogo com regras fáceis de aprender, jogadas simples de executar e um objetivo bem definido.

2.2 Objetivo

Um jogo termina quando um dos jogadores consegue ter todas as peças da sua cor agrupadas no tabuleiro ou quando o tabuleiro se encontra preenchido. Ganha o jogador que tiver o maior grupo de peças da sua cor. Em caso de empate, ganha o jogador que tiver o segundo maior aglomerado de pedras.

2.3 Equipamento

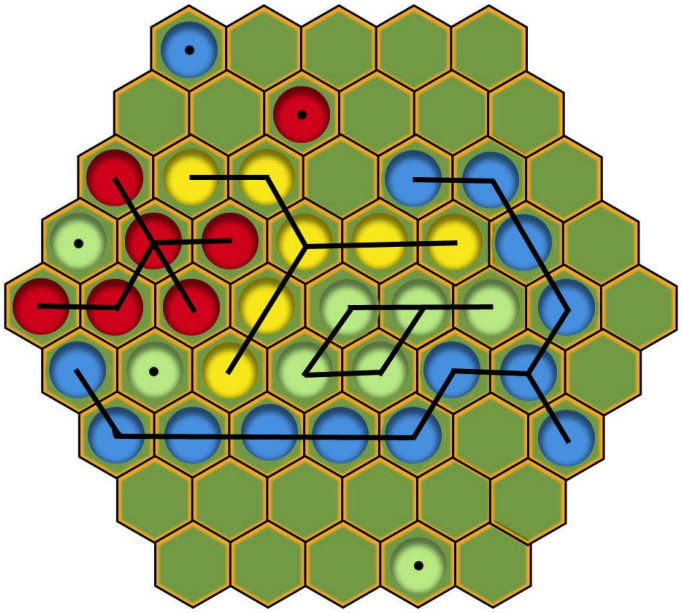
Para o jogo, será necessário um tabuleiro hexagonal, composto por hexágonos mais pequenos de cada lado, sessenta pedras com quatro cores diferentes (quinze vermelhas, quinze verdes, quinze amarelas e quinze azuis) e um saco.

2.4 Preparação

O jogo inicia-se com o tabuleiro vazio. Dentro do saco guardam-se quatro peças, uma de cada cor. Seguidamente, cada jogador deve retirar uma peça de dentro do saco, sem a revelar aos restantes. A cor da pedra retirada será a atribuída ao respetivo jogador, a qual deverá ser mantida em segredo até ao final do jogo.

2.5 Grupos

O termo ‘grupo’, neste jogo, é definido como um conjunto de peças da mesma cor que se encontram interligadas (uma única peça pode ser considerada um grupo).



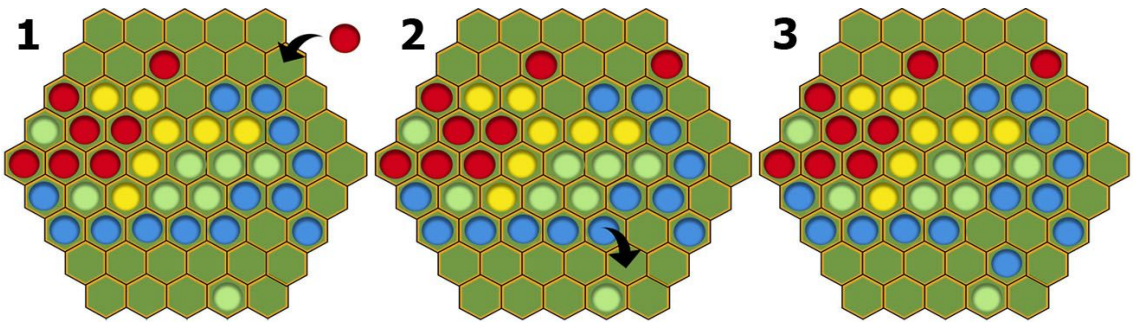
A imagem representativa do tabuleiro, acima apresentada, demonstra uma possível situação de jogo na qual se podem distingui diversos grupos de pedras das quatro cores diferentes. Dado que basta duas peças serem adjacentes para formarem um grupo, conclui-se que a cor azul possui a liderança.

2.6 Rondas

A cada ronda, os jogadores deverão executar uma jogada apenas, passando a sua vez no sentido anti-horário do tabuleiro.

2.7 Jogada

O jogador deverá depositar uma pedra de qualquer cor, retirada aleatoriamente do saco, num espaço livre do tabuleiro e de seguida deslocar uma outra peça, já em campo, para outro espaço livre.



2.8 Restrições

Uma pedra pode ser movida em linha reta em qualquer direção até que uma outra pedra se depare no seu caminho ou os limites do tabuleiro lhe bloqueiem o movimento. Será, por isso, impossível que uma pedra salte por cima de outra.

É, também, estritamente proibido que os jogadores comuniquem entre si durante o jogo.

3. Representação do Estado do Jogo

A informação do jogo será armazenada numa estrutura composta por uma lista de listas. A lista inicial representara o tabuleiro, a secundária armazenará as posições de cada fila do campo e as respetivas sub-listas apresentarão o estado de cada posição.

Tendo em atenção que o campo de jogo apresenta um formato hexagonal, com 5 unidades de cada lado, as listas secundárias serão constituídas por um número variável de elementos.

Nas figuras seguintes apresentamos os diferente estados da lista ao longo do desenrolar do jogo.

```
hexBoard(B0) :- B0 =
[   [[' '],[' '],[' '],[' '],[' ']],
    [[' '],[' '],[' '],[' '],[' '],[' ']],
    [[' '],[' '],[' '],[' '],[' '],[' '],[' ']],
    [[' '],[' '],[' '],[' '],[' '],[' '],[' '],[' ']],
    [[' '],[' '],[' '],[' '],[' '],[' '],[' '],[' '],[' ']],
    [[' '],[' '],[' '],[' '],[' '],[' '],[' '],[' ']],
    [[' '],[' '],[' '],[' '],[' '],[' '],[' ']],
    [[' '],[' '],[' '],[' '],[' '],[' ']],
    [[' '],[' '],[' '],[' '],[' ']]
].
```

```
hexBoard(B0) :- B0 =
[   [['B'],[' '],[' '],[' '],[' ']],
    [[' '],[' '],['R'],[' '],[' '],[' ']],
    [['R'],['Y'],['Y'],[' '],['B'],['B'],[' ']],
    [['G'],['R'],['R'],['Y'],['Y'],['Y'],['B'],[' ']],
    [['R'],['R'],['R'],['Y'],['G'],['G'],['G'],['B'],[' ']],
    [['B'],['G'],['Y'],['G'],['G'],['B'],['B'],[' ']],
    [['B'],['B'],['B'],['B'],['B'],[' '],['B']],
    [[' '],[' '],[' '],[' '],[' '],[' ']],
    [[' '],[' '],[' '],['G'],[' ']]
].
```

Uma posição vazia no tabuleiro será representada por um espaço em branco (‘ ‘). Por sua vez, a posição ocupada por cada pedra guardará a sua respetiva cor, podendo esta ser azul(‘B’), vermelha(‘R’), verde(‘G’) ou amarela(‘Y’).

4. Visualização do Tabuleiro

O tabuleiro é representado através da interface textual do SICStus. Para o efeito, foram criadas várias funções que permitem visualizar o tabuleiro.

De forma a obter esta vizualização, realiza-se a chamada da seguinte função:

hexBoard0(Board),drawBoard(Board).

Estas funções. futuramente serão melhoradas de forma a que o tabuleiro e as várias posições possíveis das peças não sejam "hard coded", como nos exemplos abaixo.

```
hexBoard1(Board) :- Board =
[    [['B'], [' '],[' '], [' '],[' ']],
     [[' '], [' '],['R'], [' '],[' '], [' ']],
     [['R'], ['Y'],['Y'], [' '],['B'], ['B'],[' ']],
     [['G'], ['R'],['R'], ['Y'],['Y'], ['Y'],['B'], [' ']],
     [['R'], ['R'],['R'], ['Y'],['G'], ['G'],['G'], ['B'], [' ']],
     [['B'], ['G'],['Y'], ['G'],['G'], ['B'],['B'], [' ']],
     [['B'], ['B'],['B'], ['B'],['B'], [' '],['B']],
     [[' '], [' '],[' '], [' '],[' '], [' ']],
     [[' '], [' '],[' '], ['G'],[' ']]
].
```

```
hexBoard2(Board) :- Board =
[    [[' '], [' '],[' '], [' '],[' ']],
     [[' '], [' '],[' '], [' '],[' '], [' ']],
     [[' '], [' '],[' '], [' '],[' '],[' ']],
     [[' '], [' '],[' '], [' '],[' '],[' '], [' ']],
     [[' '], [' '],[' '], [' '],[' '],[' '], [' '], [' ']],
     [[' '], [' '],[' '], [' '],[' '],[' '], [' ']],
     [[' '], [' '],['B'], [' '],[' '], [' '],[' ']],
     [[' '], [' '],[' '], [' '],[' '], [' ']],
     [[' '], [' '],[' '], [' '],[' ']]
].
```

```
hexBoard0(Board) :- Board =
[    [[' '], [' '],[' '], [' '],
      [' ']],
     [[' '], [' '],[' '], [' '],
      [' '], [' ']],
     [[' '], [' '],[' '], [' '],
      [' '], [' '],[' ']],
     [[' '], [' '],[' '], [' '],
      [' '], [' '],[' '], [' ']],
     [[' '], [' '],[' '], [' '],
      [' '], [' '],[' '], [' '],
      [' ']],
     [[' '], [' '],[' '], [' '],
      [' '], [' '],[' '], [' ']],
     [[' '], [' '],[' '], [' '],
      [' '], [' ']],
     [[' '], [' '],[' '], [' '],
      [' ']]
].
```

```
hexBoard3(Board) :- Board =
[    [['B'], ['R'],['R'], ['R'],['R']],
     [['R'], ['R'],['R'], ['R'],[' '], [' ']],
     [['R'], ['Y'],['Y'], ['R'],['B'], ['B'],[' ']],
     [['G'], ['R'],['R'], ['Y'],['Y'], ['Y'],['B'], [' ']],
     [['R'], ['R'],['R'], ['Y'],['G'], ['G'],['G'], ['B'], [' ']],
     [['B'], ['G'],['Y'], ['G'],['G'], ['B'],['B'], [' ']],
     [['B'], ['B'],['B'], ['B'],['B'], [' '],['B']],
     [[' '], [' '],[' '], [' '],[' '], [' ']],
     [[' '], [' '],[' '], ['G'],[' ']]
].
```

```
hexBoard4(Board) :- Board =
[    [['B'], ['G'],['G'], ['Y'],['G']],
     [['G'], ['G'],['R'], ['Y'],['Y'], ['Y']],
     [['R'], ['Y'],['Y'], ['Y'],['B'], ['B'],['Y']],
     [['G'], ['R'],['R'], ['Y'],['Y'], ['Y'],['B'], ['Y']],
     [['R'], ['R'],['R'], ['Y'],['G'], ['G'],['G'], ['B'], [' ']],
     [['B'], ['G'],['Y'], ['G'],['G'], ['B'],['B'], [' ']],
     [['B'], ['B'],['B'], ['B'],['B'], ['R'],['B']],
     [['G'], ['R'],['R'], ['R'],['R'], [' ']],
     [[' '], [' '],['R'], ['G'],['R']]
].
```

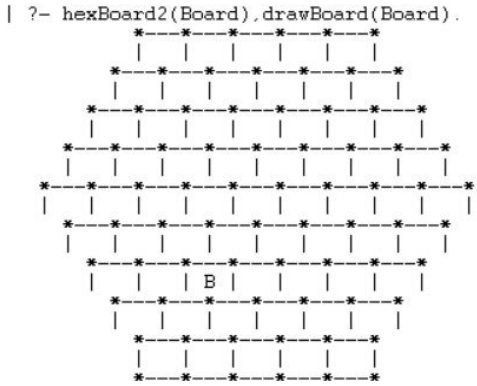
```
drawBoard(Board) :- secundaryDrawBoard(Board, 0).

secundaryDrawBoard([Line | Board], Number) :- Board \= [],
                                              drawIdentation(Number), displayBoarder(Line),
                                              drawIdentation(Number), displayLine(Line),
                                              NewNumber is Number+1,
                                              secundaryDrawBoard(Board, NewNumber).

secundaryDrawBoard([Line], Number) :-   drawIdentation(Number), displayBoarder(Line),
                                        drawIdentation(Number), displayLine(Line),
                                        drawIdentation(Number), displayBoarder(Line).

displayBoarder([_|Line]) :- write('   *---'), displayAuxBorder(Line).


displayAuxBorder([]) :- write('*\n').

displayAuxBorder([_|Line]) :- write('*---'), displayAuxBorder(Line).

writeAuxLine([]) :- write('|\n').

writeAuxLine([Elem|Line]) :-    write('|'), writeSymbol(Elem),
                                writeAuxLine(Line).

displayLine(Line) :- write(' '), write(' '), write(' '),
                     writeAuxLine(Line).


writeSymbol([Color]) :- write(' '), write(Color), write(' ').

drawIdentation(0) :- write('        ').
drawIdentation(1) :- write('      ').
drawIdentation(2) :- write('    ').
drawIdentation(3) :- write('  ').
drawIdentation(4) :- write('').
drawIdentation(5) :- write('  ').
drawIdentation(6) :- write('    ').
drawIdentation(7) :- write('      ').
drawIdentation(8) :- write('        ').
```

Para demonstrar os tabuleiros, decidimos apresentar uma situação de jogo.

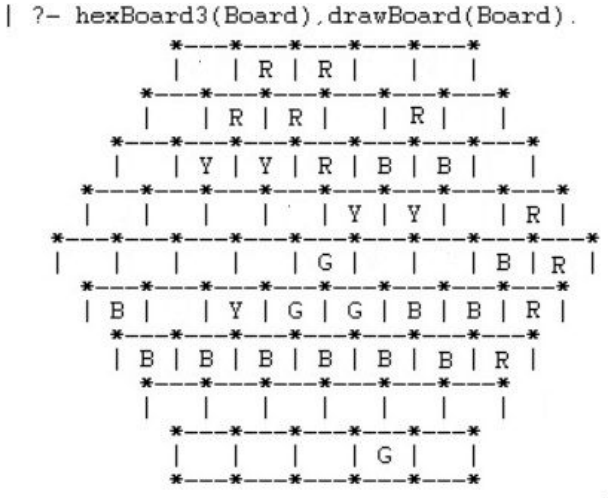
Inicialmente, como dito anteriormente, o tabuleiro encontra-se completamente vazio.

```
| ?- hexBoard(Board),drawBoard(Board).
          *---*---*---*---*---*
          |   |   |   |   |   |
        *---*---*---*---*---*---*
        |   |   |   |   |   |   |
      *---*---*---*---*---*---*---*
      |   |   |   |   |   |   |   |
    *---*---*---*---*---*---*---*---*
    |   |   |   |   |   |   |   |   |
  *---*---*---*---*---*---*---*---*---*
  |   |   |   |   |   |   |   |   |   |
    *---*---*---*---*---*---*---*---*
    |   |   |   |   |   |   |   |   |
      *---*---*---*---*---*---*---*
      |   |   |   |   |   |   |   |
        *---*---*---*---*---*---*
        |   |   |   |   |   |   |
          *---*---*---*---*---*
          |   |   |   |   |   |
            *---*---*---*---*
            |   |   |   |   |
            *---*---*---*---*
```

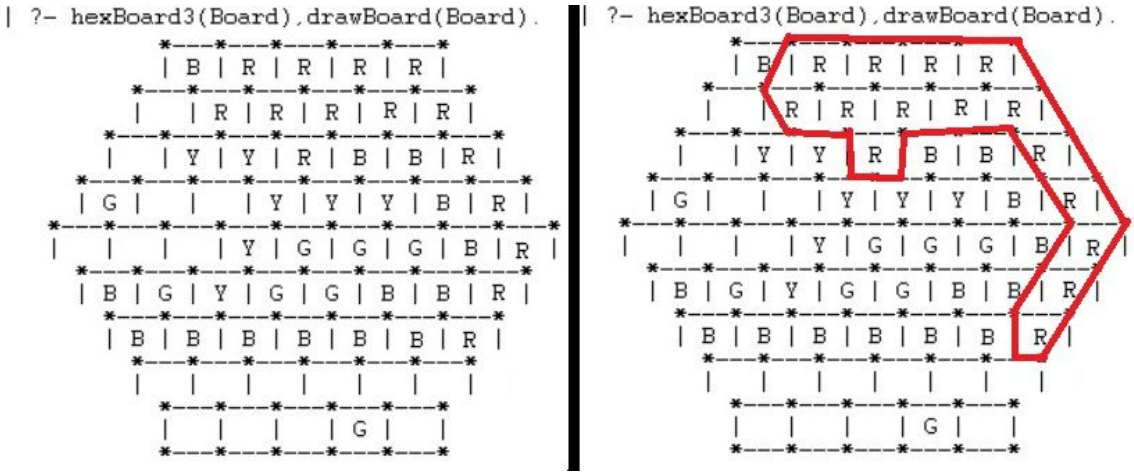
Na primeira jogada, o jogador mais novo, joga uma peça de cor aleatória no tabuleiro na posição que desejar.



Numa fase de jogo mais avançada, o tabuleiro pode tomar uma forma semelhante ao tabuleiro abaixo



A imagem abaixo representa o estado final do jogo. Neste exemplo, o jogador que sai vencedor é aquele que tem como "segredo" as pedras da cor vermelha pois conseguiu com que o grupo de maior peças da mesma cor fosse das vermelhas.



5. Movimentos

Como foi abordado nos pontos 2.7 e 2.8 deste relatório, concluímos que as possíveis jogadas que cada jogador poderá executar são bastante simples e rápidas de executar: colocar uma nova pedra num espaço livre do tabuleiro e mover uma outra pedra para outro espaço vazio, obedecendo às restrições.

Apresentamos agora, 6 cabeçalhos para os principais predicados que serão aplicados para a implementação do jogo.

```
generateStone(Board, Nr, Ng, Nb, Ny).

dropStone(Board, Row, Col).
slideStone(Board, StartRow, StartCol, FinalRow, FinalCol, FinalBoard, RoundNumber).

checkEmptyPosition(Board, Row, Col).
checkValidSlide(Board, StartRow, StartCol, FinalRow, FinalCol).

nextTurn(Board, RoundNumber).
```

6. Fontes

http://www.boardgamegeek.com/boardgame/161090/meerkats .

http://www.ludosoup.com/2014_06_01_archive.html .