Syrtis

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo 79:
Flávio Couto - 201303726
Pedro Afonso Castro - 201304205

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

10 de Outubro de 2015

1 Introdução

O principal objetivo deste projeto é adquirir competências ao nível da Programação em Lógica através do estudo de um jogo de tabuleiro previamente escolhido. Foram-nos dadas várias hipóteses de jogos, tendo nós escolhido o Syrtis, por causa da sua elevada componente estratégica e tática, bem como alguma complexidade, sempre necessária para trazer um maior sentimento de desafio quando nos é colocado um projeto em mãos.

2 O Jogo Syrtis

Segue-se uma explicação das regras do Syrtis, bem como alguns exemplos para melhor demonstrar alguns aspetos que possam ser de maior dificuldade de compreensão.

2.1 História

O Syris é um jogo de estratégia em que os dois jogadores se encontram numa ilha instável e desconhecida. A paisagem da ilha está sempre em mudança, e o avanço do mar faz com que a ilha esteja a tornar-se cada vez mais pequena... Para complicar ainda mais, este avanço está a fazer com que se formem areias movediças, limitando ainda mais o espaço habitável na ilha... Apenas um dos jogadores poderá sobreviver, quem será capaz de ser o mais forte?

2.2 Objetivo

A cada jogador é atribuída uma forma e uma cor (quadrado e preto ou círculo e branco). O objetivo do jogo é conquistar todas as peças que contenham a sua cor ou a sua forma.

2.3 Equipamento

O jogo é composto por peças quadradas com uma determinada forma e cor. Há 4 combinações possíveis: círculos e quadrados brancos e pretos. Há também quatro torres, duas brancas e circulares e duas pretas e quadradas.

2.4 Preparação

As peças quadradas são inicialmente aleatoriamente dispostas num de dois formatos de ilha. Para um jogo mais longo e estratégico, utiliza-se o formato Syrtis Major. Para um jogo mais curto e tático, utiliza-se o formato Syrtis Minor. As figuras 1 e 2 mostram a estrutura destes dois formatos.

Depois, um dos jogadores coloca as 4 torres por cima de uma das peças à sua escolha, desde que sejam da cor ou da forma dessa peça. O outro jogador decide se quer jogar com as torres brancas circulares ou pretas quadradas e o jogo começa por quem tiver ficado com as peças brancas e circulares.

2.5 Ilhas

Uma ilha é uma peça ou um grupo de peças que partilham a mesma forma ou cor. Para serem consideradas uma ilha, devem estar adjacentes horizontal ou verticalmente. Há quatro tipos de ilhas: ilhas pretas, brancas, quadradas e

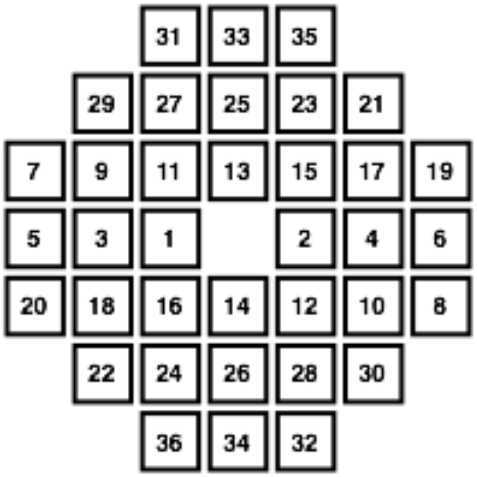

Figura 1: Syrtis Major

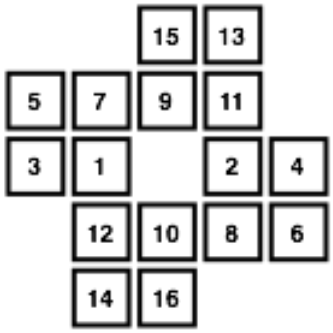

Figura 2: Syrtis Minor

circulares. Uma peça pode obviamente pertencer a uma ilha de uma cor e a uma ilha de uma forma ao mesmo tempo. As figuras 3 e 4 mostram exemplos de ilhas.

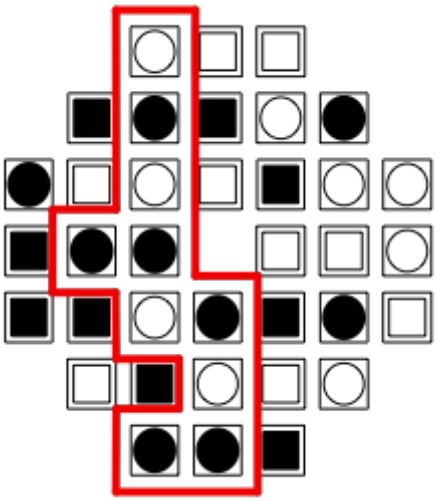

Figura 3: Ilha de círculos

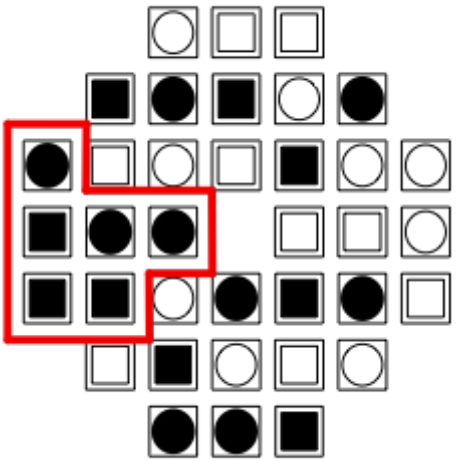

Figura 4: Ilha de pretos

2.6 Acções

Em cada turno cada jogador pode fazer uma de entre 4 acções possíveis: mover uma torre, afundar uma peça, deslocar uma peça, ou passar a sua vez.

2.6.1 Mover uma torre

Cada jogador pode mover uma das suas torres, desde que se mantenha em pelo menos uma das suas duas ilhas atuais (ou seja, ou na ilha respeitante

à forma ou na ilha respeitante à cor). As outras torres não bloqueiam este movimento, ou seja, podemos passar por cima de outras torres. A figura 5 mostra um exemplo do movimento de uma torre.

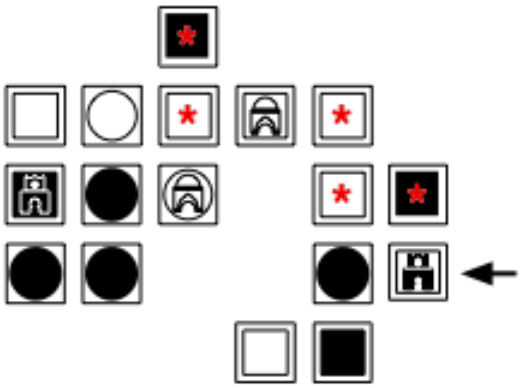

Figura 5: Movimento de uma torre

A torre indicada (preta e quadrada) pode mover-se para qualquer uma das peças marcadas com uma seta, pois estas encontram-se na sua ilha quadrangular. Note-se também que ela não pode ir para a peça à sua esquerda, visto que, apesar de ser preta, esta não se encontra na sua ilha (visto que a peça em que a torre se encontra é branca e quadrada, e a peça em questão é preta e circular).

2.6.2 Afundar uma peça

Cada jogador pode remover uma peça do tabuleiro se:

- Esta for adjacente a uma peça ocupada por uma torre desse jogador;
- Esta estiver desocupada;
- Esta tiver pelo menos um espaço adjacente livre.

A figura 6 mostra um exemplo de quais peças podem ser afundadas.

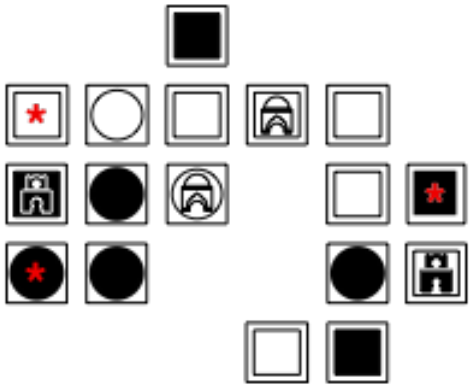

Figura 6: Peças afundáveis

O jogador preto pode afundar quaisquer peças que estejam marcadas com uma estrela. Nenhuma peça branca está numa posição que lhe permita afundar alguma peça.

2.6.3 Deslocar uma peça

Um jogador poderá deslocar uma peça ocupada por uma das suas torres através de quaisquer espaços vazios que estejam conectados. A peça pode ser movida em qualquer direcção, quantas vezes quisermos, desde que a largura ou altura do tabuleiro não seja aumentada (nem mesmo a meio do movimento) e as peças do tabuleiro continuem conectadas. A figura 7 mostra um exemplo disto.

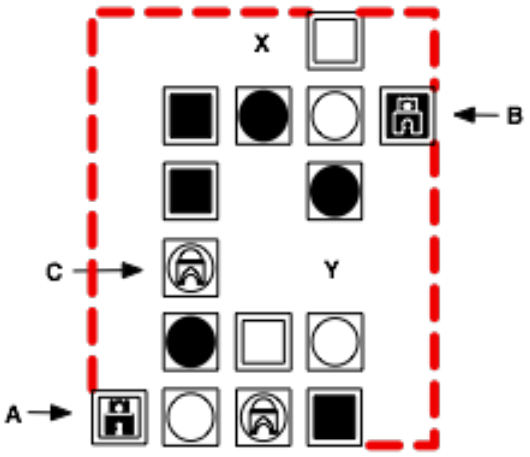

Figura 7: Peças deslocáveis

A torre A pode mover a sua peça para a posição X, mas a torre B não pode (visto que teria de aumentar o tamanho do tabuleiro a meio da jogada para o fazer). A torre C apenas se pode deslocar para a posição Y, porque é a unica posição em que o tabuleiro continua todo conectado. A outra torre não se pode mexer.

2.6.4 Passar a vez

Basta não fazer nada e passar a vez ao outro jogador. Um jogador que não pode fazer nada é obrigado a passar.

2.7 Fim do jogo

Há três formas de ganhar o jogo. Uma já foi dita anteriormente (ter uma ilha completa). As outras duas são usadas principalmente para evitar empates e que os jogos durem demasiado tempo.

2.7.1 Ilha completa

Um jogador vence quando todas as peças restantes da sua cor, ou da sua forma, estão conectadas. Se todas as peças de uma cor ou forma estão conectadas quando o tabuleiro é inicialmente construído, o jogador que colocar as torres deve inicalmente trocar duas peças cuja numeração no formato de jogo

usado (ver figuras 1 e 2) sejam seguidas. Por exemplo, trocar a peça 11 com a peça 12. Deve continuar a usar-se este método até que tal deixe de acontecer.

2.7.2 Areias movediças

Se um jogador afundar quatro peças sem o outro jogador ter afundado nenhuma, o segundo perde, sendo engolido pelas areias movediças. Os jogadores devem controlar quantas peças afundaram desde que o outro jogador afundou a última peça.

2.7.3 Iniciativa

Esta regra existe para evitar empates. O jogo termina se alguma das seguintes situações ocorrer:

- Ambos os jogadores conseguem uma ilha completa ao mesmo tempo;
- Ambos os jogadores passam a sua vez em jogadas consecutivas (ou seja, ambos passam duas vezes cada um);
- Um jogador passa quatro vezes seguidas.

Em qualquer um dos casos, o último jogador a afundar uma peça ganha. Se nenhum dos jogadores tiver afundado uma peça, o jogador que jogou primeiro ganha.

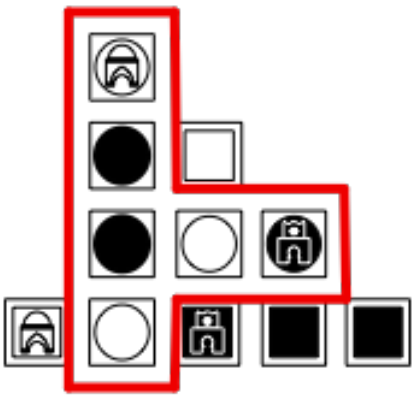

Figura 8: Uma situação em que o jogador branco ganhou ao completar uma ilha de círculos.

3 Representação do Estado do Jogo

O jogo será representado numa lista de listas de listas. As duas primeiras listas representarão o tabuleiro de jogo. A 3ª lista representará o conteúdo de cada quadrado do tabuleiro, utilizando o formato [Torre,Cor,Forma].

- **Torre:** Representa a existência ou não de uma torre no quadrado. Em caso afirmativo, utiliza a letra 'B' para representar uma torre branca, circular, e uma letra 'P' para uma torre preta, quadrangular.

- **Cor:** Representa a cor da do quadrado. Se for preto, utiliza-se a letra 'P', se for branco utiliza-se a letra 'B'.
- **Forma:** Representa a forma da peça do quadrado. Se for um círculo, usa-se a letra 'C', se for um quadrado usa-se a letra 'Q'.

Para todos os casos, a não existência do objeto implica a representação através de um espaço (' ').

4 Visualização do Tabuleiro

O tabuleiro é visto recorrendo à interface textual do SICStus. Para efeitos de demonstração, fizemos dois tabuleiros, um do tipo Syrtis Major e outro do tipo Syrtis Minor, *hard-coded*. Para os obter, faz-se uma das seguintes chamadas:

```
major_board(Board), display_board(Board).
```

ou

```
minor_board(Board), display_board(Board).
```

As funções irão futuramente criar um tabuleiro aleatório com o formato especificado nas figuras 1 e 2 respetivamente.

Código da implementação (incluindo os dois tabuleiros *hard-coded*):

```
major_board(Board) :- Board =
[       [[' ',' ',' '], [' ',' ',' '],[' ','B','C'], [' ','B','Q'],
         [' ','B','Q'], [' ',' ',' '],[' ',' ',' ']],
         [' ',' ',' '],[' ','P','Q'],[' ','P','C'], [' ','P','Q'],
         ['L','B','C'], [' ','P','C'], [' ',' ',' ']],
        [[' ','P','C'], [' ','B','Q'], ['L','B','C'], [' ','B','Q'],
         [' ','P','Q'], [' ','B','C'], [' ','B','C']],
        [[' ','P','Q'], [' ','P','C'], [' ','P','C'], [' ',' ',' '],
         ['T','B','Q'], [' ','B','Q'], [' ','B','C']],
        [[' ','P','Q'], [' ','P','Q'], [' ','B','C'], [' ','P','C'],
         [' ','P','Q'], [' ','P','C'], [' ','B','Q']],
        [[' ',' ',' '], ['T','B','Q'], [' ','P','Q'], [' ','B','C'],
         [' ','B','Q'], [' ','B','C'], [' ',' ',' ']],
        [[' ',' ',' '], [' ',' ',' '], [' ','P','C'],
         [' ','P','C'], [' ','P','Q'], [' ',' ',' '], [' ',' ',' ']]].

minor_board(Board) :- Board =
[       [[' ',' ',' '], [' ',' ',' '],[' ','B','C'], [' ','B','Q'],
         [' ','B','Q'], [' ',' ',' '],[' ',' ',' ']],
         [' ',' ',' '],[' ','P','Q'],[' ','P','C'], [' ','P','Q'],
         ['L','B','C'], [' ','P','C'], [' ',' ',' ']],
        [[' ','P','C'], [' ','B','Q'], ['L','B','C'], [' ','B','Q'],
         [' ','P','Q'], [' ','B','C'], [' ','B','C']],
        [[' ','P','Q'], [' ','P','C'], [' ','P','C'], [' ',' ',' '],
         ['T','B','Q'], [' ','B','Q'], [' ','B','C']],
        [[' ','P','Q'], [' ','P','Q'], [' ','B','C'], [' ','P','C'],
         [' ','P','Q'], [' ','P','C'], [' ','B','Q']],
        [[' ',' ',' '], ['T','B','Q'], [' ','P','Q'], [' ','B','C'],
         [' ','B','Q'], [' ','B','C'], [' ',' ',' ']],
        [[' ',' ',' '], [' ',' ',' '], [' ','P','C'],
```

```
        [' ','P','C'], [' ','P','Q'], [' ',' ',' '], [' ',' ',' ']]].

display_board(Board) :- write_col_coords(Board), display_board_aux(Board,1).

display_board_aux([Line | Board],Number) :- Board \= [], write_border(Line),
write_line(Line, Number), NextNumber is Number + 1,
display_board_aux(Board,NextNumber).

display_board_aux([Line], Number) :-write_border(Line),
write_line(Line, Number), write_border(Line).

write_border([_|Line]) :- write('   +----'), write_border_aux(Line).

write_border_aux([]) :- write('+\n').

write_border_aux([_|Line]) :- write('+----'), write_border_aux(Line).

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

write_aux_line([Elem|Line]) :- write('|'), write_elem(Elem),
write_aux_line(Line).

write_line(Line, Number) :- write(' '), write(Number), write(' '),
write_aux_line(Line).

write_elem([Tower,Colour,Shape]) :- write(Tower), write(Colour), write(Shape).

write_col_coords([Line | Board]) :- write('   '),
write_col_coords_aux(Line,65).

write_col_coords_aux([Elem],Charcode) :-        char_code(Character,Charcode),
write('  '), write(Character), write('\n').

write_col_coords_aux([Elem|Line],Charcode) :- Line \= [],
char_code(Character,Charcode), write('  '), write(Character),
write(' '), Nextchar is Charcode+1, write_col_coords_aux(Line,Nextchar).
```

A figura 9 mostra uma imagem com o output produzido.

5 Movimentos

Tal como já foi referido anteriormente na secção 2.6, existem 4 jogadas possíveis que um jogador pode fazer na sua jogada: mover uma torre, afundar uma peça, deslocar uma peça ou passar a vez. Apresentam-se 4 cabeçalhos para os predicados que serão utilizados para implementar estas acções:

```
valid_slide(Board, StartRow, StartCol, FinalRow, FinalCol).
slide_tile(Board, StartRow, StartCol, FinalRow, FinalCol, FinalBoard).

valid_remove(Board,Row,Col).
remove_tile(Board,Row,Col,Board).
```

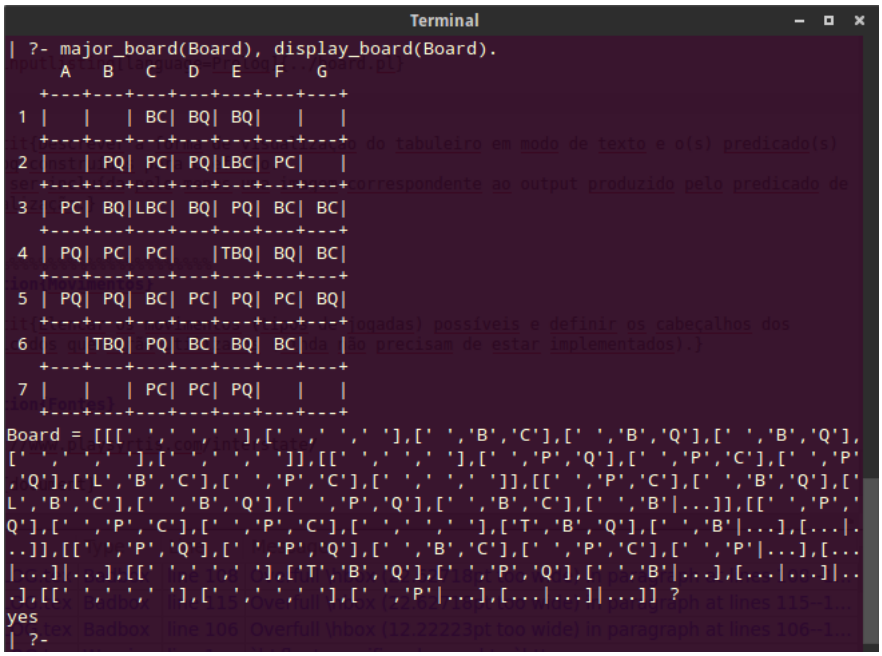

Figura 9: Output produzido pelas chamadas a majorBoard e minorBoard

```
valid_move(Board, StartRow, StartCol, FinalRow, FinalCol).
move_tower(Board, StartRow, StartCol, FinalRow, FinalCol, FinalBoard).

pass(Board).
```

6 Fontes

http://www.playsyrtis.com/interstate/