Crab Stack

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo Crab_Stack_1:
Inês de Sousa Caldas - up200904082
Maria Teresa Chaves - up201306842

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

16 de Outubro de 2016

1 O Jogo *Crab Stack*

O jogo *Crab Stack* foi publicado pela *Blue Orange Games* em 2015. Este foi conceptualizado por *Henri Kermarrec* que contou com a colaboração da artista *Stéphane Escapa*. Enquadra-se nas categorias de jogos abstratos e familiares, sendo aconselhado a jogadores com idade superior a 8 anos[1].

Este jogo pode ser jogado por dois a quatro jogadores. Dependendo do número de jogadores apenas uma parte do tabuleiro é utilizado. No caso de serem dois jogadores utiliza-se apenas as rochas amarelas (figura 1); no caso de serem três jogadores, para além das rochas amarelas também são utilizadas as rochas pretas; e por último no caso de serem quatro jogadores, joga-se em todo o tabuleiro[2]. No contexto da unidade curricular de Programação em Lógica, é pretendido o desenvolvimento de um jogo de tabuleiro para dois jogadores. Desta forma apenas serão enunciadas as regras[3] relacionadas com o modo de dois jogadores.

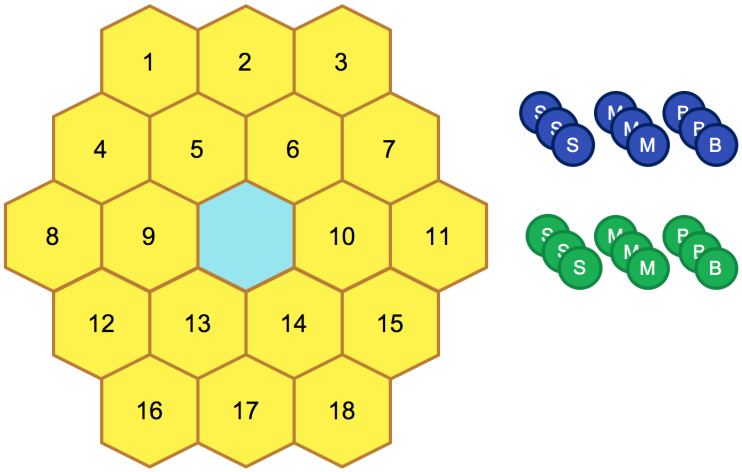


Figura 1: Tabuleiro de jogo

1.1 Preparação do tabuleiro

Com um formato hexagonal, o tabuleiro é formado por 18 posições, como é possível observar na figura 1 nas rochas amarelas.
A cada jogador é atribuído um conjunto de 9 caranguejos: 3 grandes, 3 médios e 3 pequenos, como é possível observar no lado direito da figura 1.
Antes do jogo começar, os caranguejos dos jogadores são distribuídos aleatoriamente pelo tabuleiro obedecendo às regras de empilhamento que serão explicitadas na secção 1.2.

1.2 Movimentos possíveis

No turno de um jogador, este pode mover um dos seus caranguejos de forma a terminar o seu movimento numa rocha ocupada por outro caranguejo. De-

[1] http://www.geekyhobbies.com/crab-stack-review-and-instructions/
[2] http://www.blueorangegames.com/index.php/games/crab-stack
[3] https://boardgamegeek.com/boardgame/172033/crab-stack

pendendo do seu tamanho, um caranguejo move-se, obrigatoriamente, um determinado número de rochas:

- Pequeno - move-se três rochas;
- Médio - move-se duas rochas;
- Grande - move-se uma rocha.

No final do movimento, um caranguejo não pode regressar à rocha inicial.
Apenas os caranguejos que estejam no topo da pilha podem ser movidos durante o turno de um jogador.
O empilhamento dos caranguejos tem de obedecer às seguintes regras:

- um grande pode ficar em cima de qualquer outro caranguejo;
- um médio pode ficar em cima de um caranguejo médio ou pequeno;
- um pequeno pode apenas ficar em cima de outro caranguejo pequeno.

1.3 Regra da Onda

Os caranguejos gostam de estar em grupos grandes e não gostam de ser separados. Numa situação em que os caranguejos fiquem separados em dois grupos, por uma linha de rochas vazias, um destes grupos irá ser eliminado do jogo por uma onda. O grupo a ser eliminado é selecionado pela seguinte ordem de prioridades:

1. O grupo de caranguejos que ocupar menos espaço no tabuleiro é removido do jogo;
2. Se os dois grupos ocupam o mesmo número de casas no tabuleiro, então o grupo com o menor número total de caranguejos é removido do jogo;
3. Se o número de caranguejos for o mesmo, o jogador do turno em que ocorreu a separação, decide qual o grupo que irá ser removido do jogo.

1.4 Fim do jogo

O *Crab Stack* pode terminar nos seguintes estados do jogo:

- Se todas as peças de um jogador forem removidas do jogo, através de uma onda, o jogador perde o jogo;
- Se um jogador, no inicio do seu turno, não conseguir mover nenhum dos seus caranguejos, perde o jogo;
- Caso o jogo chegue a um ponto em que os jogadores estão repetidamente a executar os mesmos movimentos, o jogo termina e ocorre um empate. Desta forma, é necessário realizar um novo jogo para determinar o vencedor do jogo.

2 Representação do Estado do Jogo

2.1 Representação do estado do tabuleiro

O tabuleiro é representado por uma lista de listas, em que cada elemento da lista representa uma rocha. Os caranguejos de cada jogador serão representados por Sx, Mx, Bx, em que x representa o número do jogador e S, M, B o tamanho do caranguejo, respetivamente pequeno, médio e grande. Uma rocha, i.e. uma célula da lista, tem dois estados diferentes possíveis: estar vazia ou ter uma pilha de caranguejos. Uma pilha de caranguejos, pode ser por exemplo [B1, S1, S2], em que o elemento B1 representa o topo da pilha.
O código para a criação de um tabuleiro é dado por:

```
make_board(-Board, +NumPlayers):-
        NumPlayers == 2,
        % for other number of players, only the number of lines
        % and  the size change
        Size is 3,
        Line is 5,
        make_board_aux(-Board,[], +Line, +Size).
```

2.2 Posições iniciais do jogo

No início do jogo, as peças de cada jogador são colocadas aleatoriamente no tabuleiro, garantindo que fica com todas as rochas ocupadas. Abaixo segue-se o predicado que inicia o jogo e um exemplo de um tabuleiro no estado inicial do jogo.

```
init_crabs(-FinalBoard, +Board, +Crabs).
Board =  [ %initial board example
                [[S1][M2][M1]]
                [[B2][B1][M1][S2]]
                [[S2][S1][W][M2][B2]]
                [[M2][B2][S1][B1]]
                [[M1][S2][B1]]
        ]
```

Para uma melhor visualização do estado inicial representado acima, segue-se uma figura ilustrativa.

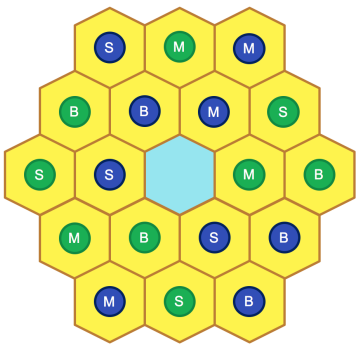


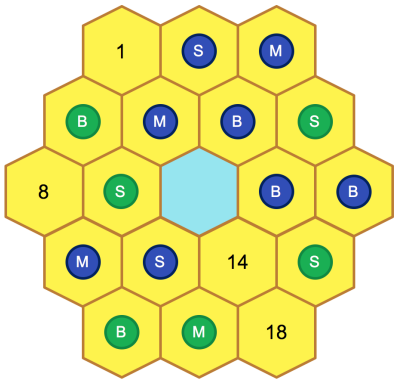
Figura 2: Estado inicial do jogo

2.3 Posições intermédias do jogo

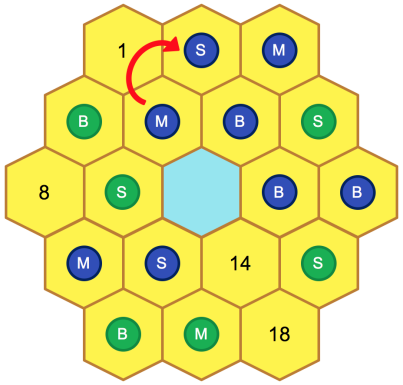
Abaixo seguem-se exemplos de tabuleiros num estado intermédio do jogo.

```
Board =  [%General Intermediate state board
                [[M1,M2,S1][][]]
                [[][B1][M1][S2]]
                [[B2,S2][S1][W][M2][B1,B2]]
                [[B2,M2][][B1,S1][]]
                [[][M1,S2][]]
         ]

Board =  [% Intermediate state board With a possibility for a wave
                [[][S1][M1]]
                [[B2][M1][B1][S2]]
                [[][S2][W][B1,M2][B1,B2]]
                [[M1,M2][S1][][S2,S1]]
                [[B2][M2][]]
         ]

Board =  [% Intermediate state board ready for the Wave
                [[][M1,S1][M1]]
                [[B2][][B1][S2]]
                [[][S2][W][B1,M2][B1,B2]]
                [[M1,M2][S1][][S2,S1]]
                [[B2][M2][]]
         ]

Board =  [% Intermediate state board after the Wave
                [[][M1,S1][M1]]
                [[][][B1][S2]]
                [[][][W][B1,M2][B1,B2]]
                [[][][][S2,S1]]
                [[][][]]
         ]
```

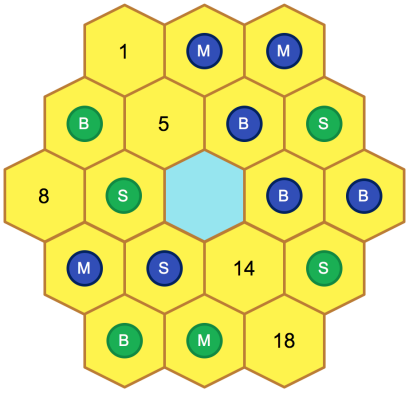
Para uma melhor visualização da regra da onda, seguem-se imagens ilustrativas do exemplo apresentado acima, com os diferentes estados intermédios.



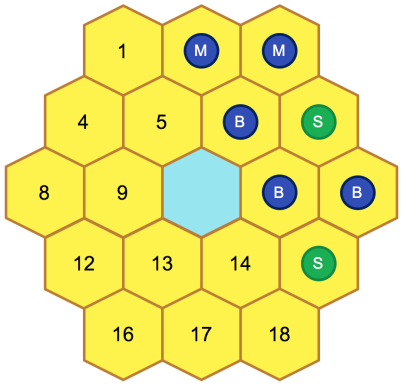
(a) Possibilidade de ocorrer uma onda.



(b) Jogador azul move peça para ocorrer uma onda.



(c) Ocorre uma onda.



(d) Peças do grupo esquerdo são removidas do jogo.

(e) O jogador azul ganha o jogo.

Figura 3: Exemplo da ocorrência de uma onda durante o turno do jogador azul.

2.4 Posições finais do jogo

No exemplo da onda, mencionada na secção anterior, verifica-se que após a onda, o jogador verde apenas pode mover um dos seus caranguejos pequenos para cima do seu outro caranguejo pequeno, deixando de ter movimentos possíveis, e portanto o jogador azul ganha o jogo.

```
Board =  [% Green's turn after the wave
            [[][M1,S1][M1]]
            [[][][B1][S2]]
            [[][][W][B1,M2][B1,B2]]
            [[][][][S2,S1]]
            [[][][]]
         ]

Board =  [% Final state board after the green's move
            [[][M1,S1][M1]]
            [[][][B1][S2,S2]]
            [[][][W][B1,M2][B1,B2]]
            [[][][][S1]]
            [[][][]]
         ]
```

3 Visualização do Tabuleiro

Para a visualização do tabuleiro, irão ser implementados os seguintes predicados:

- Cabeçalho do predicado para a visualização do tabuleiro:

  `display_board(+Board).`

- Cabeçalho do predicado para a visualização da pilha:

  `display_stack(+Rock, +Stack).`

Segue-se a visualização do output do predicado *display_board* para o primeiro exemplo de um estado intermédio de jogo apresentado na secção 2.3.

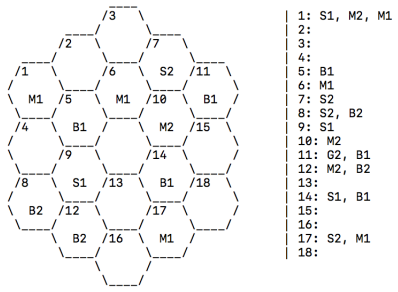


Figura 4: Representação do output no estado intermédio do jogo.

4 Movimentos

Para os movimentos existentes no *Crab Stack*, explicados na secção 1.2, serão implementados os seguintes predicados:

- Cabeçalho do predicado para o movimento de um caranguejo:

  `moveCrab(-FinalBoard, +Board, +CrabSize, +PosInit, +PosFinal).`

- Cabeçalho do predicado para a verificação da ocorrência de uma onda:

  `checkWave(-FinalBoard, +Board).`

- Cabeçalho do predicado para a verificação do fim do jogo, i.e., a impossibilidade de movimentos por parte de um dos jogadores:

  `checkMoves(-Winner, +Board).`