Barragoon

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo Barragoon_3:

Tiago Carvalho - up201504461

Rui Quaresma - up201503005

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

12 de Novembro de 2017

Resumo

No decorrer do semestre foi desenvolvido, em grupo, um jogo em PROLOG, denominado de Barragoon, que consiste num jogo de tabuleiro para dois jogadores.

Desde cedo, tornou-se evidente que esta linguagem de programação é significativamente diferente das que estávamos habituados até então, sendo assim este o obstáculo mais interessante de superar e um apesto fulcral na motivação do grupo para concluir o trabalho.

Quanto à apresentação do jogo, foram encontrados alguns desafios, nomeadamente devido ao extenso número de peças/faces diferentes, para as quais tivemos de definir uma representação clara e simples. Aquando do desenvolvimento da lógica surgiram várias dificuldades, designadamente na verificação do número de jogadas válidas para averiguar o final do jogo, e também na crição do bot, sendo este o problema que necessitou de mais tempo para ser resolvido.

Apesar de se terem erguido várias adversidades, foram todas ultrapassadas com sucesso e ficamos satisfeitos com o resultado final do projeto.

Conteúdo

1 Introdução

A escolha deste jogo como tema do projeto foi baseada na complexidade do, sendo que procurámos desenvolver um jogo cujas regras sejam de fácil entendimento, no entanto que permita um elevado número de jogadas, de modo a tornar-se muito estratégico.

O relatório encontra-se dividido em três grandes partes, informações sobre o jogo em si, implementção da lógica em PROLOG e finalmente a interface com o utilizador.

2 O Jogo Barragoon

Criado em 2014, Barragoon é um fascinante jogo estratégico de informação perfeita sem qualquer fator de sorte. Inicialmente as 14 peças e 8 ”barragoons” são colocadas no tabuleiro da seguinte forma (Figura 1), ficando de lado as restantes 24 ”barragoons”.

Cada jogador move as suas 7 peças de modo a reposicionar as ”barragoons” e conquistar todas as peças do seu adversário, finalizando assim uma partida. A peça ”barragoon” é o elemento central deste jogo e tem como objetivo bloquear ou libertar movimentos em direções específicas, dependendo do símbolo voltado para cima.

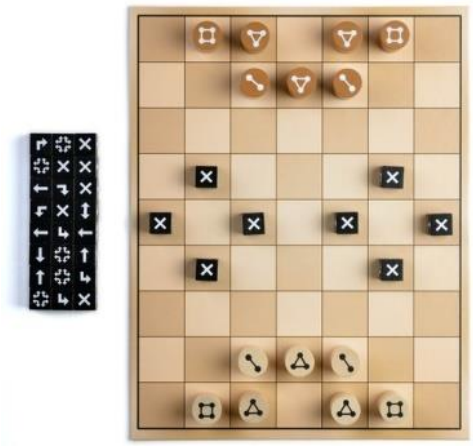

Figura 1: Tabuleiro inicial

Quando uma ”barragoon” é capturada, o jogador que efetuou a captura pode escolher uma nova posição para a mesma e que face fica voltada para cima. Para além das ”barragoons” há mais 3 tipos de peças diferentes:

- “2-space tile”
- “3-space tile”
- “4-space tile”

Cada uma destas peças pode-se mover apenas na horizontal ou na vertical. Durante um movimento, a direção pode ser apenas alterada uma vez e sempre num ângulo de 90º. Não é possível passar por cima das próprias peças ou das do adversário, nem capturar as próprias peças.

Existem dois movimentos possíveis:

- "Full move" – a peça move-se o máximo de casas possíveis, respetivamente 2, 3 e 4
- "Short move" – a peça move-se o máximo de casas possíveis menos uma, respetivamente 1, 2 e 3

Apenas é possível capturar uma peço do adversário ou uma "barragoon" num "full move".

3 Lógica do Jogo

3.1 Representação do Estado do Jogo

O tabuleiro é constituído por uma lista de listas com os seguintes elementos:

- 00 – Casa vazia
- 22 – Peça castanha de movimento máximo 2
- 23 – Peça castanha de movimento máximo 3
- 24 – Peça castanha de movimento máximo 4
- 12 – Peça branca de movimento máximo 2
- 13 – Peça branca de movimento máximo 3
- 14 – Peça branca de movimento máximo 4
- 30 – Barragoon em cruz
- 31 – Barragoon em linha reta de baixo para cima
- 32 – Barragoon em linha reta da esquerda para a direita
- 33 – Barragoon em linha reta da direita para a esquerda
- 34 – Barragoon em linha reta de cima para baixo
- 35 – Barragoon em linha reta de baixo para cima ou cima para baixo
- 36 – Barragoon em linha reta da esquerda para a direita ou da direita para a esquerda
- 37 – Barragoon "all turns", com curvas de todas as direções
- 40 – Barragoon em curva de baixo para a direita
- 41 – Barragoon em curva da esquerda para baixo
- 42 – Barragoon em curva de cima para a esquerda
- 43 – Barragoon em curva da direita para cima
- 44 – Barragoon em curva de baixo para a esquerda
- 45 – Barragoon em curva da direita para baixo
- 46 – Barragoon em curva de cima para a direita
- 47 – Barragoon em curva da esquerda para cima

Inicialmente o tabuleiro apresenta-se neste estado:

initialBoard([

[00, 24, 23, 00, 23, 24, 00],

[00, 00, 22, 23, 22, 00, 00],

[00, 00, 00, 00, 00, 00, 00],

[00, 30, 00, 00, 00, 30, 00],

[30, 00, 30, 00, 30, 00, 30],

[00, 30, 00, 00, 00, 30, 00],

[00, 00, 00, 00, 00, 00, 00],

[00, 00, 12, 13, 12, 00, 00],

[00, 14, 13, 00, 13, 14, 00]]).

Figura 2: Estado Inicial

Com o decorrer do jogo, peças vão sendo capturadas e "barragoons" vão sendo adicionadas, podendo o tabuleiro se apresentar como no seguinte exemplo:

intBoard([

[00, 00, 00, 00, 00, 00, 00],

[00, 00, 00, 00, 00, 00, 00],

[00, 00, 00, 00, 00, 00, 30],

[00, 00, 30, 00, 34, 31, 14],

[31, 24, 00, 34, 30, 31, 33],

[12, 34, 30, 31, 00, 24, 31],

[30, 31, 00, 33, 35, 30, 00],

[31, 35, 35, 00, 00, 33, 00],

[22, 13, 00, 33, 00, 00, 00]]).

Figura 3: Estado intermédio

O jogo termina quando um dos jogadores captura as sete peças do seu adversário:

finalBoard([

[00, 00, 00, 00, 00, 00, 00],

[00, 00, 00, 00, 00, 00, 00],

[30, 32, 00, 33, 00, 00, 30],

[35, 00, 30, 00, 34, 31, 00],

[31, 24, 00, 34, 30, 31, 33],

[00, 34, 30, 31, 00, 24, 31],

[30, 31, 00, 33, 35, 30, 00],

[31, 35, 35, 35, 00, 33, 00],

[22, 00, 00, 33, 00, 31, 00]]).

3.2 Visualização do Tabuleiro

Código usado para apresentar o tabuleiro na consola:

```
printBoard(Board) :-

printCoordsX,

printBlackLine,

printRowByRow(Board, 1).


printRowByRow([], N).


printRowByRow([Line|Rest], N) :-

write(' |'),

printSingleRowFirst(Line),

write(N),

write('|'),

printSingleRowSecond(Line),

write(' |'),

printSingleRowThird(Line),

N1 is N+1,

printRowByRow(Rest, N1).
```

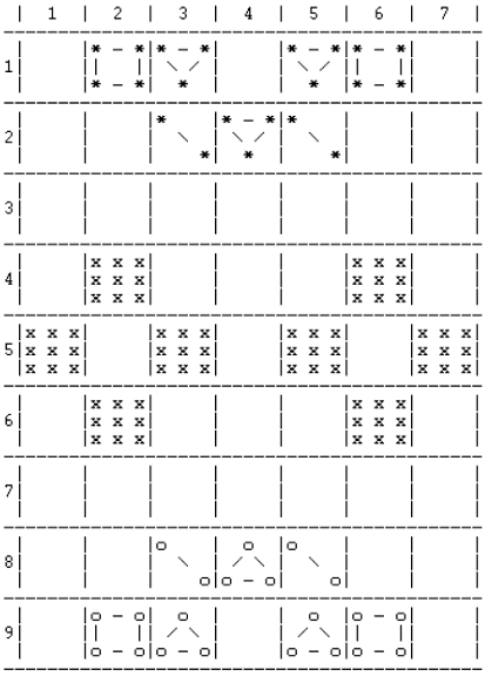

Figura 4: Visualização na consola

3.3 Lista de Jogadas Válidas

```
% Predicate that is used to validate a move.
% First it retrieves the piece from the given coordinates for it.
% Then it verifies if the new coordinates contain a piece that belongs to the current player.
% After that, it retrieves the whatever the new coordinates contain and calls the predicate
% verifyNewPosInRange and the predicate verifyPieceInTheWay.

validateMove(Board, Line, Column, NewLine, NewColumn, Player):-
  getPiece(Board, Line, Column, Piece),
  checkNewPosPiece(Board, NewLine, NewColumn, Player),
  getPiece(Board, NewLine, NewColumn, NewPiece),
  !,
  verifyNewPosInRange(NewPiece, Piece, Line, Column, NewLine, NewColumn),
  !,
  verifyPieceInTheWay(Line, Column, NewLine, NewColumn, Board).
```

Figura 5: Predicado que valida uma jogada

3.4 Execução de Jogadas

```
 % Predicate that applies the move of the selected piece to the new position, putting 00 in the original coordinates
 % and the Piece in the new coordinates. It also verifies whether the new coordinates had a piece there,
 % If not, the predicate returns true.
 % If they had, the board, after the moves, is printed and it is called a predicate to check the type of piece it was.

move(Board, NewBoard3, Line, Column, NewLine, NewColumn, Mode, AiLevel):-
   getPiece(Board, Line, Column, Piece),
   getPiece(Board, NewLine, NewColumn, NewPiece),
   updateBoard(Line, Column, 00, Board, NewBoard),
   updateBoard(NewLine, NewColumn, Piece, NewBoard, NewBoard2),
   (((NewPiece =\= 00), !, printBoard(NewBoard2), !, nl, verifyTypeNewPiece(NewPiece, NewBoard2, NewBoard3, Mode, AiLevel));
    NewBoard3 = NewBoard2).
```

Figura 6: Predicado que realiza o movimento de uma peça

3.5 Avaliação do Tabuleiro

```
% Predicate that uses the setof predicate to get the valid moves for the given player.
% It also sets a value to each valid move accordingly.
% But in this case it is checked if the place of destination is one of the oponnents
% destination as well. If it is, this destination is descarted.

getValuedValidMovesAvoidCapture(Board, ValidMoves, Player):-
    ((Player == 1, P1 is 2) ; (Player == 2, P1 is 1)),
    (setof([[Value, NewLine, NewCol], [PieceLine, PieceCol]],(getPlayerPieces(Board, Player, PieceCol, PieceLine),
    validateMoveNoMessages(Board, PieceLine, PieceCol, NewLine, NewCol, Player),
    checkMoveForInCheckPlace(Board, P1, NewLine, NewCol),
    setMoveValue(Board, NewLine, NewCol, Value)),
    ValidMoves) ; true).

% Predicate that uses the setof predicate to get the valid moves for the given player.
% It also sets a value to each valid move accordingly.

getValuedValidMovesNormal(Board, ValidMoves, Player):-
    (setof([[Value, NewLine, NewCol], [PieceLine, PieceCol]],(getPlayerPieces(Board, Player, PieceCol, PieceLine),
    validateMoveNoMessages(Board, PieceLine, PieceCol, NewLine, NewCol, Player),
    setMoveValue(Board, NewLine, NewCol, Value)),
    ValidMoves) ; true).
```

```
% Predicate that gives a certain move a value, according to the piece present in the destination.

setMoveValue(Board, NewLine, NewCol, Value):-
    getPiece(Board, NewLine, NewCol, Piece),
    (((Piece == 00), Value is 0);
     ((((Piece >= 30) , (Piece =< 37)) ; ((Piece >= 40) , (Piece =< 47))), Value is 1);
     ((Piece == 12 ; Piece == 22), Value is 2);
     ((Piece == 13 ; Piece == 23), Value is 3);
     ((Piece == 14 ; Piece == 24), Value is 4)).
```

Figura 7 e 8: Predicados que criam as listas com jogadas válidas, ordenadas por valor

3.6 Final do Jogo

```
% Predicate used for game end verification.
% First it checks if any of the players has any pieces left, if so, the game continues,
% otherwise the game ends. Then it gets each player valid moves and checks if
% any of those is empty, if not, the game continues, otherwise the game ends.

verifyEndGame(Board):-
  (((whiteCounter(WC), (WC == 0),!, nl, write('Brown won'), nl, nl) ;
  (brownCounter(BC), (BC == 0),!, nl, write('White won'), nl, nl)),
  !,fail);
  player(P),
  ((P==1, P1 is 2); (P==2, P1 is 1)),
  (getValidMoves(Board, ValidMovesPlayer, P) ; true),
  length(ValidMovesPlayer, NumValidMovesPl),
  (getValidMoves(Board, ValidMovesOp, P1) ; true),
  length(ValidMovesOp, NumValidMovesOp),
  (((NumValidMovesPl == 0 ; NumValidMovesOp == 0),
  ((P == 1, ((NumValidMovesPl == 0, nl, write('Brown won')) ;
  (NumValidMovesOp == 0, nl, write('White won'))), nl, nl) ;
  (P == 2, ((NumValidMovesPl == 0, nl, write('White won')) ;
  (NumValidMovesOp == 0, nl, write('Brown won'))), nl, nl)), !, fail) ; true).
```

Figura 9: Predicado que verifica se o jogo terminou

3.7 Jogada do Computador

```
% Predicate that is called when the AiLevel is 1 and starts with showing in the screen
% which player turn is it.
% Then gets the player valid moves, checks if there is more than one and if there is,
% it creates a random number between 1 and the number of valid moves, getting the
% correspondent piece coordinates and new coordinates to move it to.
% If there is only ine valid move, it gets the coordinates from that.

getAiPlay(Board, Line, Column, NewLine, NewColumn, 1):-
  tellPlayerToPlay(_),
  player(P),
  getValidMoves(Board, ValidMoves, P),
  length(ValidMoves, NumValidMoves),
  ((NumValidMoves == 1, PosValidMoves is NumValidMoves);
  (NumValidMoves =\= 1, random(1, NumValidMoves, PosValidMoves))),
  nth1(PosValidMoves, ValidMoves, Move),
  nth1(1, Move, NewPlace),
  nth1(2, Move, Piece),
  nth1(1, Piece, Line),
  nth1(2, Piece, Column),
  nth1(1, NewPlace, NewLine),
  nth1(2, NewPlace, NewColumn).
```

```
% Predicate that is called when the AiLevel is 2 or 3 and starts with showing in the screen
% which player turn is it.
% Then gets the player valued valid moves, checks if there is more than one and if there is,
% it then checks if the most valuable play has a value equal to 0 and if it does,
% it creates a random number between 1 and the number of valid moves, getting the
% correspondent piece coordinates and new coordinates to move it to.
% If the most valuable play has a value greater than 0, then this will be the chosen move.
% If there is only ine valid move, it gets the coordinates from that.

getAiPlay(Board, Line, Column, NewLine, NewColumn, _):-
    tellPlayerToPlay(_),
    player(P),
    getValuedValidMoves(Board, ValidMoves, P),
    length(ValidMoves, NumValidMoves),
    ((NumValidMoves == 1, nth1(NumValidMoves, ValidMoves, Move));
    nth1(NumValidMoves, ValidMoves, MoveCheck),
    nth1(1, MoveCheck, NewPlaceCheck),
    nth1(1, NewPlaceCheck, Value),
    ((Value == 0,
      random(1, NumValidMoves, RandomMoveNum),
      nth1(RandomMoveNum, ValidMoves, Move));
      nth1(NumValidMoves, ValidMoves, Move)
    )),
    nth1(1, Move, NewPlace),
    nth1(1, Move, NewPlace),
    nth1(1, NewPlace, Value),
    nth1(2, Move, Piece),
    nth1(1, Piece, Line),
    nth1(2, Piece, Column),
    nth1(2, NewPlace, NewLine),
    nth1(3, NewPlace, NewColumn).
```

Figuras 10 e 11: Predicados que escolhe as jogadas dos bots, dependendo do nível da dificuldade

4 Interface com o Utilizador

O projeto está dividido em quatro ficheiros sendo que no board.pl estão todos os predicados para fazer display do board e no game.pl estão os predicados para fazer display aos diversos menus.

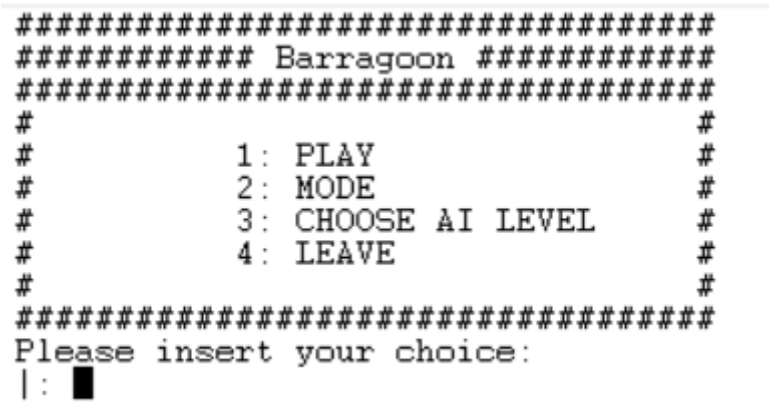

Figura 12: Menu inicial

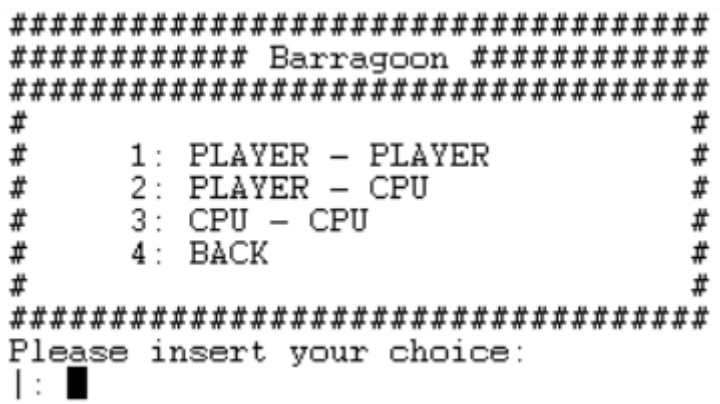

Figura 13: Menu para escolha do modo de jogo

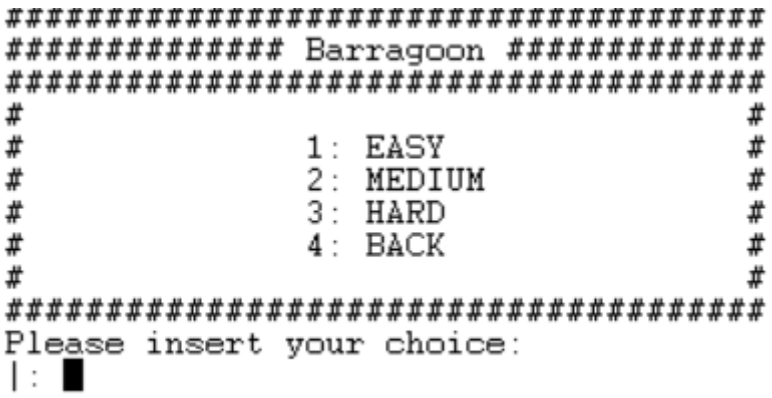

Figura 14: Menu para escolha do nível do CPU

5 Conclusões

O trabalho desenvolvido no âmbito da unidade curricular de Programação em Lógica foi uma ótima forma de aprender uma nova linguagem de programação, especialmente uma tão diferente das que até agora conhecíamos.

Se tivéssemos mais algum tempo, gostaríamos de ter melhorado o bot "hard", de modo a que este se tornasse realmente dificil de derrotar. Contudo achamos que o resultado final é bastante positivo.

Em suma, o grupo ficou bastante satisfeito tanto com o resultado como com a experiência que foi desenvolver um jogo em PROLOG.

6 Bibliografia

- http://www.barragoon.de
- https://boardgamegeek.com/boardgame/157779/barragoon

7 Anexos

A totalidade do código desenvolvido encontra-se dentro da pasta zip, juntamente com este relatório.