Ploy

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo Ploy_4:
Gonçalo Ribeiro - 201403877
Nuno Martins - 201405079

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

13 de Novembro de 2016

Resumo

Para o 1º Trabalho Prático, o nosso grupo implementou o jogo de tabuleiro Ploy em Prolog - uma linguagem de programação em lógica. O principal objectivo deste trabalho foi aplicar os conhecimentos adquiridos sobre o paradigma de programação em lógica, lecionados na unidade curricular PLOG.

O resultado final é um versão funcional do jogo com interface de texto. Foi também implementada um inteligência artificial capaz de fazer jogadas aleatórias ou gananciosas.

Como considerações finais, estamos satisfeitos com o trabalho que realizámos e concordamos que foi uma experiência enriquecedora e que de certa forma colocou à prova os nossos conhecimentos de lógica e a nossa capacidade de aprender um novo paradigma de programação num espaço de tempo relativamente pequeno.

Conteúdo

1 Introdução

No âmbito da unidade curricular de Programação em Lógica, foi-nos proposto para o primeiro trabalho prático a elaboração de um jogo de tabuleiro com a línguagem Prolog. O jogo foi selecionado de uma lista de jogos sedida pelos docentes. A escolha do Ploy como objecto do nosso trabalho residiu no interesse mútuo que temos por jogos da família do xadrez.

O objetivo deste trabalho foi pôr em prática os conhecimentos adquiridos sobre programação em lógica. Programação em lógica é um paradigma de programação baseado em lógica formal. Um programa escrito neste paragima é um conjunto de premissas e expressões em forma lógica, que expressam factos e regras sobre o dominio do problema. A línguagem Polog é uma das linguagens de programação lógica, desenvolvida por Alain Colmerauer - cientista de computação francês - em 1972.

2 O Jogo Ploy

Ploy é um jogo de tabuleiro para 2 ou 4 jogadores. Neste trabalho apenas nos focámos na versão de 2 jogadores.

Lançado em 1970 pela a empresa Minnesota Mining and Manufacturing, o jogo consiste num tabuleiro 9 por 9 onde cada jogador tem um conjunto de 15 peças. O objetivo do jogo é capturar o Commander adversário (semelhante ao Rei no xadrez) ou então todas as peças menos o Commander. A captura de peças efectua-se da mesma maneira que no xadrez - movendo-se um peça para uma casa ocupada por uma peça adversária.

Em cada turno, o jogador pode escolher uma peça para jogar. Uma jogada consiste em rodar a peça escolhida (45º para qualquer sentido) ou então mover a peça numa das direções para qual ela aponta. O número de casas a mover é determinado pelo tipo de peça.

Cada conjunto de peças contem:

Propriedades \ Peças	Shields(*)	Probes	Lances	Commander
Quantidade	3	5	6	1
Casas a mover	1	2	3	1
Direções apontadas	1	2	3	4

Tabela 1: (*) Os Shields são as únicas peças que podem ser rodadas e movidas na mesma jogada.

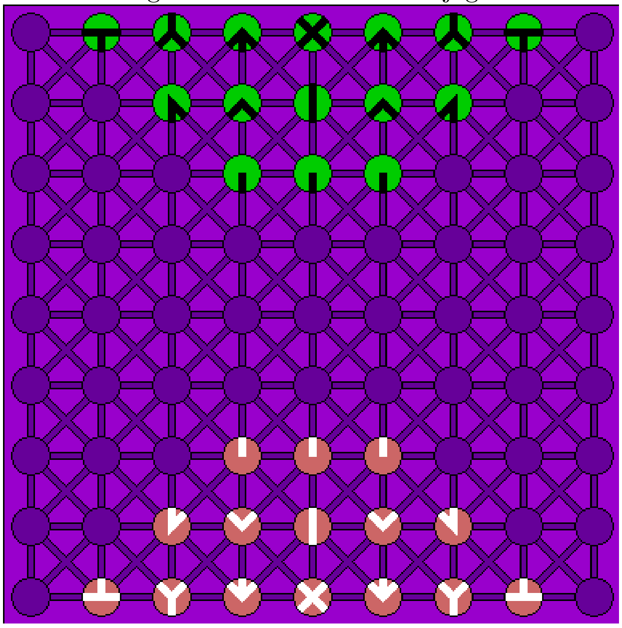
3 Lógica do Jogo

3.1 Representação do Estado do Jogo

O jogo é representado por uma lista. Dentro desta lista, existem 9 listas que representam as 9 linhas do tabuleiro. Dentro de cada uma destas últimas, 9 listas adicionais que representam as 9 quadriculas de cada linha. Dentro de cada quadricula, existe o nome da equipa ocupante - 'red', 'green' ou 'empty', caso esteja desocupada - e uma lista de pontos cardeais para qual a peça ocupante aponta (esta lista é vazia caso não exista peça).

Exemplo da estrutura do jogo:

Figura 1: Estado inicial do jogo.



```
%%%%%%%%%%%%%%%%%%%%%%%
%% Tabuleiro de jogo %%
%%%%%%%%%%%%%%%%%%%%%%%
board(
[
[*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*],
[*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*],
[*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*],
[*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*],
[*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*],
[*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*],
[*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*],
[*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*],
[*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*,*casa*]
]
).
```

Exemplo da estrutura de uma *casa*:

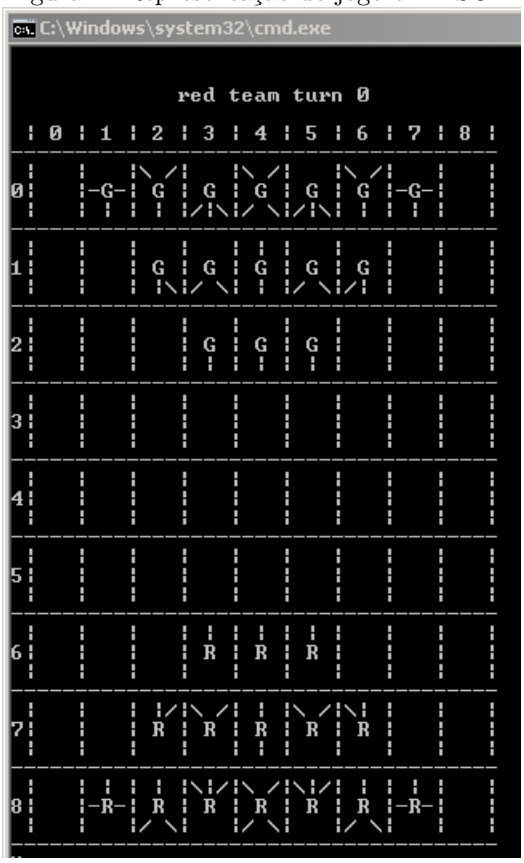
```
%%%%%%%%%%%%%%%%%%%%%%
%% Quadricula vazia %%
%%%%%%%%%%%%%%%%%%%%%%
['empty',[]]

%%%%%%%%%%%%%%%%%%%%%
%% Commander verde %%
%%%%%%%%%%%%%%%%%%%%%
['green',['sw','se','ne','nw']]
```

3.2 Visualização do Tabuleiro

A visualização do tabuleiro é feita através do predicado *draw_board(Tab)*. Neste predicado, Tab representa a estrutura de jogo supracitada - uma lista de listas. O predicado escreve a representação ASCII do tabuleiro no output buffer da consola em questão. Cada peça é representada no centro com o caracter inicial da cor correspondente (Red ou Green) e também com os respectivos indicadores de direção.

Figura 2: Representação do jogo em ASCII.



3.3 Execução de Jogadas

A realização de jogadas é feita através dos predicados:

- *movePiece(+X,+Y,+Orientation,+Length,+Board,-NewBoard,-Consumed)*
- *rotatePiece(+Piece,+Orientation,-PieceNova)*
- *setPiece(+X,+Y,+Board,-NewBoard,-NewPiece)*

Estes predicados retornam todos novo estado de jogo - NewBoard.

O predicado *movePiece* também retorna o número de direções da peça consumida (0 se nao foi consumida nenhuma), o que permite identificar o tipo de peça capturada. Neste predicado, são efetuadas várias verificações quanto à possibilidade de movimento da peça, nomeadamente:

- Se a Orientation está presente na peça - *assertHasOrientation(Orientation,Piece)*;
- Se a distância é possível - *assertValidLength(Length,Piece)*;
- Se o destino está dentro do tabuleiro - *assertInsideBoundaries(Xf,Yf)*;
- Se todas as casas até ao destino estão vazias, e se o destino ou está vazio ou contem um inimigo - *assertNoCollision(X,Y,Orientation,Length,Board)*.

A rotação de uma peça é efetuada através de *rotatePiece* e *setPiece*, não sendo necessárias nenhumas verificações, pois a rotação de uma peça nunca pode falhar.

3.4 Final do Jogo

O predicado *assertGameEnded(+Board,-Winner)* faz a verificação do final do jogo. Neste predicado, Board representa o estado de jogo atual, e Winner refere a equipa vitoriosa caso o predicado seja verdadeiro.

Por sua vez , este predicado chamará outros 2 predicados para cada equipa, *assertCommanderDead(+Board,+Team)* e *assertAllSmallDead(+Board,+Team)*, que testam as duas condições de terminação de jogo, que são, respetivamente, quando o Commander de uma das equipas é capturado ou quando todas as peças excepto o Commander de uma equipa são capturadas.

3.5 Jogada do Computador

O predicado *bot_plays_diff(Dif,Board,Team,NewBoard)* faz uma jogada automática. Neste predicado, Dif representa a dificuldade do bot, Board é o estado de jogo atual, Team a equipa do bot e NewBoard o novo estado de jogo depois de efetuada a jogada. Dif pode tomar 2 valores possíveis, podendo estes ser 0 (bot aleatório) e 1 (bot ganancioso).

O bot aleatório fará sempre jogadas completamente aleatórias, enquanto que o bot ganancioso procurará recursivamente qual a melhor jogada a fazer nesse mesmo turno, sendo o factor a maximizar a peça inimiga capturada(quantas mais direçoes tiver a peça, maior a prioridade). Assim, uma peça com 4 direções tem prioridade sobre uma de 3, esta tem prioridade sobre uma de 2, e a de 2 tem prioridade sobre a de 1.

Se não for encontrada nenhuma jogada que resulte na captura de uma peça, o bot ganancioso chama o predicado do bot aleatório, ou seja, com Dif = 0.

4 Interface com o Utilizador

Ao correr o programa, o utilizador é deparado com o menu principal.

O menu tem três opções: 'Play', 'How to play' e 'Exit'.

A opção 'Play' imprime os modos de jogo disponiveis.

Uma vez escolhido um modo de jogo onde o utilizador intervine (Player Vs. Player e Player Vs. AI), o utilizador terá a opção de fazer uma jogada. A jogada é constituida pela escolha de uma peça, seguida pela escolha entre mover ou rodar essa peça (ou ambas caso a peça seja um Shield).

A opção 'How to play' imprime as instruções do jogo no ecrã e remete novamente para o menu inicial.

Figura 3: Menu principal.

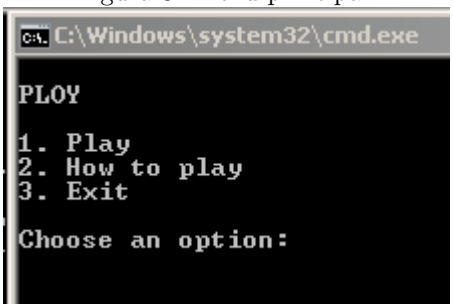


Figura 4: Menu 'Play'.

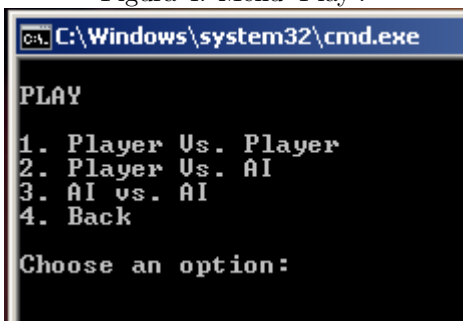


Figura 5: Um turno do jogador vermelho (*input* em baixo).

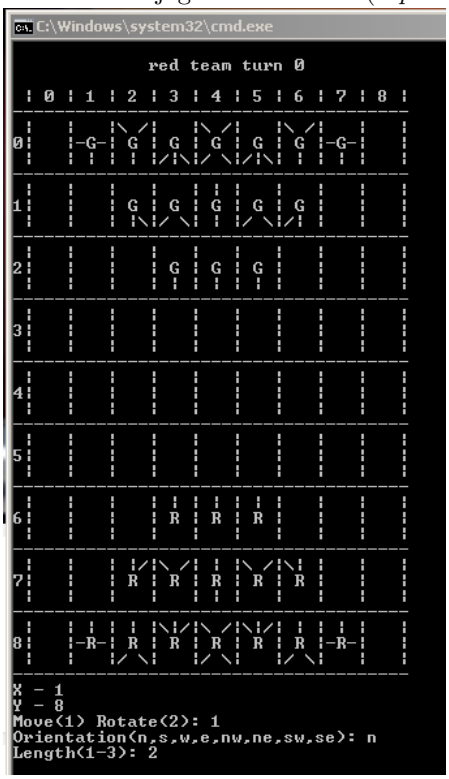


Figura 6: Turno do jogador verde.

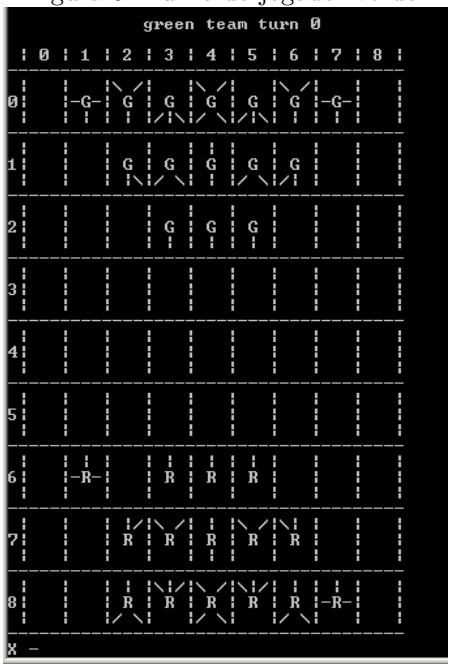
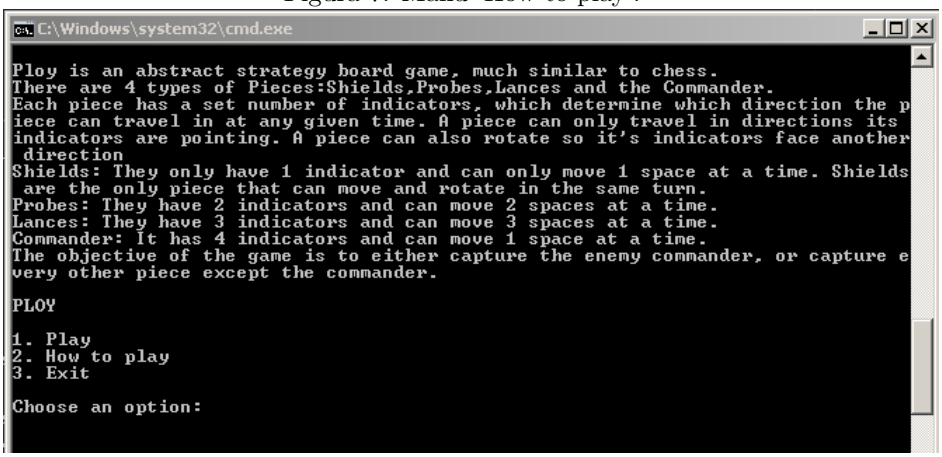


Figura 7: Manu 'How to play'.



5 Conclusões

Como considerações finais, estamos satisfeitos com o trabalho que realizámos e concordamos que foi uma experiência enriquecedora e que de certa forma colocou à prova os nossos conhecimentos de lógica e a nossa capacidade de aprender um novo paradigma de programação num espaço de tempo curto.

Não nos deparamos com nenhuma dificuldade concreta durante a implementação do jogo. No entanto, esta foi relativamente lenta e laborosa, visto que estavamos activamente a aprender Polog enquanto faziamos o projecto, o que implicou uma constante consulta de documentação e uma abordagem "trial by error"de programação.

Ambos concordamos que deviamos ter começado a elaboração do relatório mais cedo, em vez de o deixarmos para ultima prioridade. Assim teríamos mais tempo para adicionar mais conteúdo e refinar o restante. Tirando isso, sentimos que conseguimos fazer tudo ao que nos propomos no início do desenvolvimento.

Em suma, reconhecemos o potencial do Prolog como ferramenta para o ensino didatico de lógica, mas, na prespectiva de quem usa maioritariamente linguagens imperativas (C, C++, Java), é pouco funcional - não nos dá ferramentas suficientes para atacarmos diretamente um problema, é necessário criar múltiplos predicados para relizar coisas tão simples como iterar listas.