

Lear

Relatório Final



Mestrado Integrado em Engenharia Informática e
Computação

Programação em Lógica

Grupo 3:

Alexandre José da Silva Carvalho - up201506688
Vitor Emanuel Fernandes Magalhães - up201503447

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

12 de Novembro de 2017

Resumo

Resumo sucinto do trabalho com 150 a 250 palavras (problema abordado, objetivo, como foi o problema resolvido/abordado, principais resultados e conclusões).

O objetivo do trabalho, proposto pelos docentes, envolve o desenvolvimento de um jogo de tabuleiro usando PROLOG, uma linguagem lógica de programação e a implementação de uma inteligência artificial básica. O jogo de tabuleiro escolhido para este trabalho foi o Lear.

No início, foram feitos testes à linguagem de programação, utilizando predicados simples, como *initialBoard* e *getPiece*. Após a familiarização da linguagem, a lógica do jogo foi pensada e implementada.

Para a captura de peças, foi desenhada e implementada uma máquina de estados que permite lidar com os três tipos de captura. Após esta, a identificação do vencedor ficou completa.

Finalmente, foi feito uma inteligência artificial simples, com dois níveis de dificuldade. No primeiro nível, o **nível 1**, o computador joga aleatoriamente no tabuleiro. No segundo nível, o **nível 2**, o computador foi programado de maneira a calcular a melhor jogada, tendo em conta o estado atual do tabuleiro e as jogadas futuras.

O fim deste trabalho resulta num jogo simples, mas que exige concentração e bom planeamento das jogadas.

Conclui-se que PROLOG é uma linguagem lógica de capacidade rápida que contém imensos casos de uso devido à sua simplicidade e a sua expressividade.

Conteúdo

1	Introdução	4
2	O Jogo Lear	5
2.1	Regras	5
3	Lógica do Jogo	6
3.1	Representação do Estado do Jogo	6
3.2	Visualização do Tabuleiro	7
3.3	Lista de Jogadas Válidas	8
3.4	Execução de Jogadas	8
3.5	Avaliação do Tabuleiro	8
3.6	Final do Jogo	8
3.7	Jogada do Computador	8
4	Interface com o Utilizador	9
5	Conclusões	10
	Bibliografia	11

1 Introdução

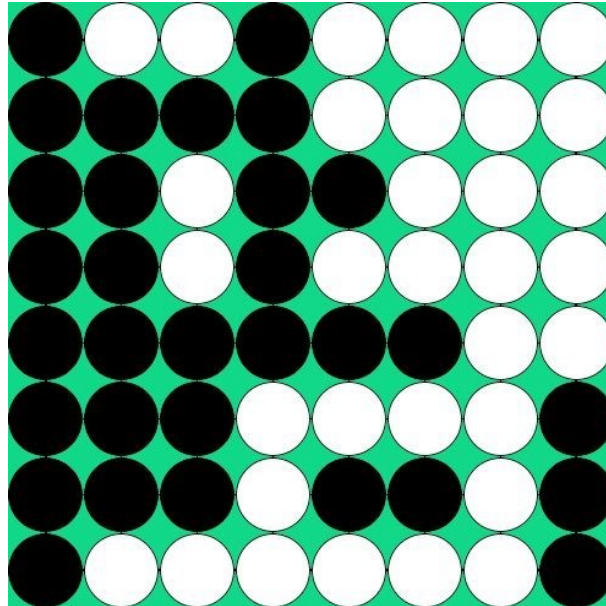
Os principais objetivos deste trabalho foram o desenvolvimento de um jogo e o reconhecimento do potencial da linguagem utilizada.

Este documento foi dividido em várias secções por forma a percorrer os seguintes tópicos:

- **Introdução** - Descrição dos objetivos e estrutura do relatório.
- **O jogo Lear** - Descrição sucinta do jogo, história e regras.
- **Lógica do jogo** - Descrição do projeto e da implementação da lógica, assim como a representação e visualização do jogo, as jogadas possíveis e avaliação destas.
- **Interface com o utilizador** - Descrição do módulo de interface com o utilizador em modo de texto.
- **Conclusões** - Síntese da informação apresentada nas secções anteriores e reflexão sobre os objetivos de aprendizagem alcançados.
- **Bibliografia** - Links utilizados.
- **Anexos** - Código fonte.

2 O Jogo Lear

Enfastiado pela obrigação de capturar peças no jogo Othello e inspirado neste último, Luís Bolaños Mures decidiu criar o Lear.



2.1 Regras

O jogo é jogado por duas equipas, a equipa Preta e a equipa Branca. A equipa preta joga primeiro.

Em cada turno, o jogador tem de colocar uma peça da sua cor num espaço vazio. Se, ao colocar a peça, constituir uma linha não interrompida de peças que contém: **a)** duas peças da sua cor e **b)** uma linha ininterrupta de peças inimigas, as peças inimigas viram, ou seja, são substituídas por peças da cor do jogador a jogar. As duas peças da mesma cor mencionadas poderão estar no início da linha ou no fim da linha, tanto juntas como uma em cada lado.

O jogo termina quando o tabuleiro estiver cheio. O vencedor é o jogador com mais pontos. Os pontos são calculados pelo número de peças mais o valor do komi, usado quando apropriado.

O komi é um valor adicionado ao número de pontos do jogador que não fez a última jogada. Este valor depende do tamanho do tabuleiro. Se o tabuleiro é ímpar, o komi será par e adicionado à equipa Branca. Caso contrário, será ímpar e adicionado à equipa Preta. O valor do komi é indicado pelo primeiro jogador. O segundo jogador escolhe a sua cor.

Na implementação deste trabalho, é apenas utilizado um Board de tamanho 64×64 e o komi foi retirado.

3 Lógica do Jogo

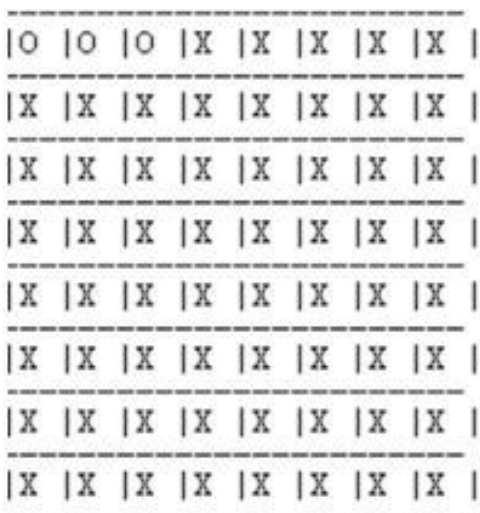
A estruturação e implementação do jogo foram feitas em várias etapas, sendo estas enumeradas pelas seguintes subsecções:

3.1 Representação do Estado do Jogo

O estado de jogo é representado por uma lista de listas. As peças Brancas são representadas por um 'O ' e as peças Pretas por um 'X '.



Figura 1: Tabuleiro Inicial



	O		O		X		X		X		X		X	
	X		X		X		X		X		X		X	
	X		X		X		X		X		X		X	
	X		X		X		X		X		X		X	
	X		X		X		X		X		X		X	
	X		X		X		X		X		X		X	
	X		X		X		X		X		X		X	
	X		X		X		X		X		X		X	

Figura 2: Tabuleiro Final, onde a Equipa Preta ganhou

3.2 Visualização do Tabuleiro

Para criar um tabuleiro, é usado o predicado `initialBoard(Board)`, como demonstrado abaixo:

```
initialBoard([
  [emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
  [emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
  [emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
  [emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
  [emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
  [emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
  [emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell],
  [emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell, emptyCell]]).
```

Figura 3: O valor `emptyCell` representa uma célula vazia.

Para imprimir o tabuleiro no ecrã, são utilizados os seguintes predicados:

```
%-----PRINT BOARD-----

printBoard(Board):-printRowSeparator, printBoardAux(Board).

printBoardAux([]).
printBoardAux([Head|Tail]) :-
    write('|'),
    printRow(Head),
    printBoardAux(Tail).

printRow([]) :- nl.
printRow([emptyCell|Tail]) :-
    write(' '),
    write('|'),
    printRow(Tail).

printRow([Head|Tail]) :-
    Head \= emptyCell,
    write(Head),
    write('|'),
    printRow(Tail).

printLineSeparator(1):- write('|'), nl.
printLineSeparator(NLines):-
    write('|'),
    nl,
    Next is NLines-1,
    printLineSeparator(Next).

printRowSeparator:-
    write('-----'), nl.
```

3.3 Lista de Jogadas Válidas

O programa não cria uma lista de jogadas possíveis. De facto, ele gera jogadas possíveis e verifica a validade de cada e armazenando a melhor possível, tendo em conta os melhores resultados para capturar peças.

3.4 Execução de Jogadas

Em cada turno, a verificação e execução de uma jogada é feita através do predicado *move(Board, Player, FinalBoard)*.

Este predicado tem três fases:

A primeira corresponde ao pedido da linha e coluna onde o jogador deseja colocar a sua peça, usando o predicado *getCoordsFromUser(-NLine, -NCol)*.

De seguida, o predicado *check(+Board, +NLine, +NCol, -NextBoard, +Player)* verifica se é possível colocar a peça nas coordenadas indicadas. Caso falhe, o predicado *move* é repetido.

Finalmente, o predicado *verifyRule(+NextBoard, +NLine, +NCol, +Player, -FinalBoard)* é chamado para tentar capturar as peças do adversário.

Este predicado foi desenvolvido utilizando uma máquina de estados. Esta máquina itera a lista dada e vai adicionado, a uma lista auxiliar, o índice correspondente ao valor da lista a ser analisado. Caso a máquina volte ao estado inicial, esta lista auxiliar é limpa. Quando a máquina atingir um dos três estados finais, o predicado *capturePiecesOnList(-Player)* é utilizado para substituir as peças a serem capturadas na lista dada.

O termo *FinalBoard* será o próximo estado de jogo.

3.5 Avaliação do Tabuleiro

O tabuleiro é atualizado após cada jogada. Desta forma, cada utilizador terá sempre a versão mais recente do tabuleiro, podendo visualizá-lo para escolher a sua próxima jogada, transmitindo as coordenadas da célula onde quer jogar.

3.6 Final do Jogo

Como o jogo termina quando o tabuleiro estiver cheio e este contém sessenta e quatro células, existe um contador *Counter* que é inicializado ao valor mencionado e é decrementado após cada jogada.

Quando atingir o valor 0, verificado no predicado *endGame(+Count)*, este último utiliza o predicado *checkWinner(+FBoard)*, que conta o número de peças da Equipa Preta e da Equipa Branca e identifica quem teve mais peças ou se houve um empate.

3.7 Jogada do Computador

O predicado *project(Board, CurrPlayer, MovesForward, NLine, NCol, First, FirstLine, FirstCol)* lida com a jogada feita pelo computador, tendo em conta o nível de dificuldade.

4 Interface com o Utilizador

A interface é feita com recurso a menus que indicam as várias opções do utilizador.

O Menu Inicial contém duas opções: Jogar uma partida ou sair do programa.

```
-----
                        LEAR
1. Play a match
2. Exit
-----
Choose an option(without the dot):
.
```

Se seleccionar *Play a Match*, será apresentado outro menu que contém os vários modos de jogo.

```
-----
                        Game Mode
1. Player vs. Player
2. Player vs. Computer
3. Computer vs. Computer
4. Back
-----
Choose an option(without the dot):
.
```

Caso o modo de jogo contenha o computador como jogador, será apresentada a opção de dificuldade da inteligência artificial.

```
-----
                        AI LEVEL
1. Random Plays
2. Checks 2 plays ahead
3. Back
-----
Choose an option(without the dot):
.
```

Ao seleccionar o modo de jogo preferido, o menu desaparece, dando lugar ao jogo em si.

5 Conclusões

Este projeto foi um bom partido para o início da utilização de programação em lógica, assim como as capacidades desta.

A simplicidade e os cálculos rápidos permitem uma compreensão rápida da linguagem e uma implementação bastante simples de um jogo de tabuleiro.

O seu uso também se estende para Inteligência Artificial, revelando o quão importante este campo é para o desenvolvimento humano.

Devido ao limite de tempo de entrega, juntamente com o tempo de aprendizagem de uma nova linguagem, o trabalho teve algumas dificuldades durante o seu desenvolvimento, nomeadamente a máquina de estados criada e a inteligência artificial.

Bibliografia

<https://boardgamegeek.com/thread/1633900/new-game-lear>
<https://boardgamegeek.com/boardgame/209777/lear>