

FACULDADE DE ENGENHARIA
DA UNIVERSIDADE DO PORTO



RELATÓRIO INTERCALAR

PROGRAMAÇÃO EM LÓGICA

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA E
COMPUTAÇÃO

Corrida de Reis

Autores:

Afonso Jorge Ramos

João Dias Conde Azevedo

up201506239@fe.up.pt

up201503256@fe.up.pt

12th October 2017

Conteúdo

1	Corrida de Reis	2
2	Representação do Estado do Jogo	3
3	Visualização do Tabuleiro	4
4	Movimentos	6
5	Bibliography	6

1 Corrida de Reis

Esta variante do xadrez tradicional, Corrida de Reis, inventada por Vernon R. Parton em 1961, tem por objectivo levar o próprio rei até à última linha antes do adversário.

Vernon Rylands Parton foi um entusiasta de xadrez e um inventor prolífico de variantes para o mesmo, sendo o Xadrez de Alice a variante por ele criada mais conhecida. Muitas das variantes por ele inventadas possuíam inspiração de personagens fictícias e histórias dos trabalhos de Lewis Carroll. Parton, tal como Lewis Carroll, dedicou grande parte da sua vida académica à matemática, mas possuía interesses vários na ciência e era um forte apoiante de Esperanto.

Já face ao jogo em causa, Corrida de Reis, cada jogador começa o jogo com todas as peças normalmente usadas no xadrez exceto os peões, ou seja, começa com 1 rei, 1 rainha, 2 torres, 2 bispos e 2 cavalos. Todas as peças (brancas e pretas) são colocadas nas primeiras duas linhas do tabuleiro e ambos os jogadores veem o jogo da mesma perspectiva. Assim, o tabuleiro inicial tem o aspeto especificado na figura 1.

Como já referido, o objectivo é ser o primeiro a levar o próprio rei até à última linha (linha 8), usando as regras do Xadrez tradicional para mover e capturar as peças.

Contudo, impoem-se restrições adicionais, listadas abaixo:

- Não é permitido atacar o rei adversário, isto é, não se podem efetuar jogadas que coloquem o rei adversário em cheque;
- Um rei não pode mover-se para uma casa coberta por uma peça adversária.

Visto que para alcançar a vitória, um jogador deve mover o seu rei para a última linha, o jogador correspondente às peças brancas possuiria uma vantagem clara, por começar primeiro. Assim, quando é o rei branco que chega primeiro à última linha, o jogador preto tem uma ronda extra para que, caso consiga colocar o seu rei na última linha nessa jogada, declara-se um empate. Desta forma compensa-se a vantagem que as brancas têm por jogarem primeiro.

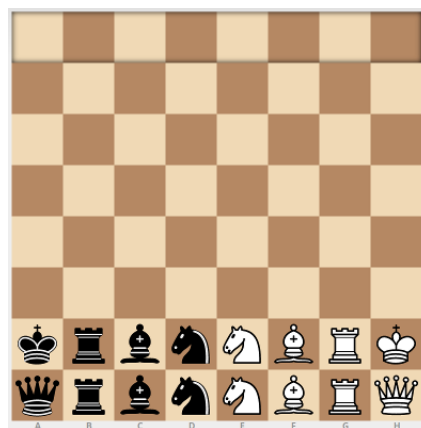


Figura 1: Tabuleiro base.

2 Representação do Estado do Jogo

O estado de jogo é guardado no tabuleiro, representado por uma lista de listas. O tabuleiro é de 8x8 e, por isso, a primeira lista conterá outras 8, cada uma dessas com 8 elementos (peças).

Para exemplificação, o código e comentários abaixo representam em linguagem PRO-LOG as posições iniciais, alguns estados intermédios e um possível final. Cada número de representação interna ao programa é traduzido em um ou mais caracteres na consola do SICStus. A chave de tradução é também apresentada em baixo.

```

(* Starting game board */
initialBoard ([[0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0],
               [6,8,9,10,5,4,3,1],
               [7,8,9,10,5,4,3,2]]).

```

```

(* Possible mid game board */
midgameBoard ([[0,0,0,0,0,0,0,0],
                [0,0,0,0,0,0,0,0],
                [0,0,0,0,0,0,0,0],
                [6,0,0,0,0,0,0,1],
                [0,8,10,0,0,0,0,0],
                [0,0,0,0,0,0,3,4],
                [0,0,9,0,5,4,0,0],
                [7,8,9,10,5,0,3,2]]).

```

```

(* Possible end game board */
endgameBoard ([[6,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,1],
               [7,0,0,0,0,0,0,2],
               [0,8,0,0,0,5,3,0],
               [0,0,0,9,0,0,0,0],
               [0,8,0,10,0,4,0,0],
               [0,0,9,10,5,4,3,0]]).

```

3 Visualização do Tabuleiro

A representação interna do tabuleiro inicial será a anteriormente apresentada, traduzindo-se no seguinte output de consola no SICStus:

8		
7		
6		
5		
4		
3		
2		BK		BT		BB		Bk		Wk		WB		WT		WK	
1		BQ		BT		BB		Bk		Wk		WB		WT		WQ	
		A		B		C		D		E		F		G		H	

Figura 2: Layout de jogo inicial.

Uma representação interna possível do tabuleiro a meio do jogo como a acima apresentada traduz-se no seguinte output de consola no SICStus:

8		
7		
6		
5		BK			WK	
4		..		BT		Bk		
3			WT		WB	
2			BB		..		Wk		WB		
1		BQ		BT		BB		Bk		Wk		..		WT		WQ	
		A		B		C		D		E		F		G		H	

Figura 3: Layout de jogo a meio.

Ainda, uma possível representação do tabuleiro num estado de jogo final será a seguinte:

8		BK		
7		
6			WK	
5		BQ			WQ	
4		..		BT			Wk		WT		..	
3			BB		
2		..		BT		..		Bk		..		WB		
1			BB		Bk		Wk		WB		WT		..	
		A		B		C		D		E		F		G		H	

Figura 4: Layout de jogo final.

```

/* Associates each number
with a chess piece */
translate(0,T) :- T = '..'.
translate(1,T) :- T = 'WK'.
translate(2,T) :- T = 'WQ'.
translate(3,T) :- T = 'WT'.
translate(4,T) :- T = 'WB'.
translate(5,T) :- T = 'Wk'.
translate(6,T) :- T = 'BK'.
translate(7,T) :- T = 'BQ'.
translate(8,T) :- T = 'BT'.
translate(9,T) :- T = 'BB'.
translate(10,T) :- T = 'Bk'.

/* Recursive function to
print current board state */
printBoard([],[]) :-
    write(' -----'), nl,
    write(' A B C D E F G H ').

printBoard([Line|Board],
           [LineNumb|Remainder]) :-
    write(' -----'), nl,
    write(LineNumb), write(' '),
    printLine(Line),
    write('| '), nl,
    printBoard(Board,Remainder).

/* Recursive function to
print each board's line */
printLine([]).
printLine([Head|Tail]) :-
    translate(Head,T),
    write('| '),
    write(T),
    printLine(Tail).

```

O tabuleiro inicial de jogo é criado usando o predicado `initialBoard(X)` em que `X` contem o tabuleiro inicial.

Para efeitos de apresentação foi construído o predicado `printBoard(X)` que recebe uma matriz `X` (lista de listas) e a imprime na consola.

É um predicado recursivo, que se auxilia noutro predicado, também ele recursivo, `printLine(Y)`. que recebe uma matriz `Y` de elementos a imprimir na consola.

Assim, é passado ao predicado `printBoard(X)`. o tabuleiro de jogo a imprimir. O mesmo separa a matriz na notação `[H—T]` em que 'H' representa a cabeça da lista (head) e 'T' a cauda da lista (tail). A cabeça apresenta-se como uma lista com os elementos da linha a imprimir, sendo passada ao `printLine(X)`. . A cauda assume-se como uma lista de listas, sendo passada novamente (chamada recursiva) ao predicado `printBoard(X)`. . Cada linha é processada recursivamente, dividida em `[H—T]`, sendo que agora a cabeça da lista representa um elemento que é traduzido usando a chave referida e impresso na consola. A restante linha assume-se como uma lista de elementos a imprimir, sendo feita uma chamada recursiva a `printLine(X)`. . Ambos os predicados apresentam como caso base o processamento de uma lista vazia.

Para efeitos de simplificação em termos de chamada na consola criou-se o predicado `printBoard`. de aridade 1 que efetua a chamada `initialBoard(X)`, `printBoard(X)`. .

4 Movimentos

Cabeçalho do predicado de movimentação de uma peça:

```
movePiece(Row, Column, EndRow, EndColumn, Board)
```

Cabeçalho do predicado de captura de uma peça:

```
deletePiece(Row, Column, Board)
```

Para qualquer um destes predicados ser válido nenhuma das peças deve colocar o rei o inimigo em cheque ou que o próprio rei em cheque:

5 Bibliography

- [1] Lichess
- [2] Chess Variants
- [3] Wikipedia
- [4] Brain King