Faculdade de Engenharia da Universidade Do Porto

Mestrado Integrado em Engenharia Informática e Computação

Programacão em Lógica

Relatório Intercalar

*Alunos:*

João Gama Amaral

João Nuno Ferreira

16 de Outubro, 2018



Índice:

* O Jogo: Knight Line
* Representação do Estado do Jogo
* Visualização do Tabuleiro
* Movimentos

**O Jogo: Knight Line**

Knight Line é um jogo de conexão minimalista para dois jogadores.

No início do jogo, cada jogador tem 20 azulejos quadrados de cor preta ou branca. Estas peças são colocadas em duas pilhas adjacentes.

Cada jogada consiste em mover uma parte da pilha (deve ser deixado para trás, no mínimo, um tijolo) para um espaço vazio fazendo um “knight’s move” (movimento baseado na peça cavalo do xadrez). Todas as peças têm de estar conectadas pelo menos diagonalmente durante o correr do jogo. O primeiro jogador a fazer uma linha de 4 ortogonalmente ou na diagonal, ganha.

**Preparação:**

Inicialmente, o jogo começa no centro do tabuleiro, colocando as duas pilhas de 20 azulejos em duas posições adjacentes, horizontalmente.

**Objetivo:**

O objetivo é conseguir fazer uma linha de 4, podendo cada elemento da linha ser uma única peça ou uma pilha com mais de uma.

**Desenvolvimento:**

Em cada jogada, cada jogador pode movimentar uma porção da pilha inicial, deixando na posição inicial, pelo menos, uma peça. As jogadas têm de respeitar o movimento “knight’s move”, porém a jogada só pode ser feita para uma posição adjacente a outra peça, quer seja uma peça do jogador ou do adversário.

**Fim:**

O fim do jogo acontece quando um jogador conseguir fazer uma linha de 4 com as suas respetivas peças, respeitando sempre o “knight’s move” e nunca deixando espaços vazios em sítios onde já jogou.

**Fonte:** <https://nestorgames.com/#knightline_detail>

**Representação do Estado do Jogo**

**Situação Inicial:**

InitialBoard([

[empty, empty, empty, empty, empty, empty],

[empty, empty, empty, empty, empty, empty],

[empty, empty, **black**, **white**, empty, empty],

[empty, empty, empty, empty, empty, empty],

[empty, empty, empty, empty, empty, empty],

[empty, empty, empty, empty, empty, empty]

]).

**Situação Intermédia:**

midBoard([

[empty, empty, empty, empty, empty, empty],

[**black**, **white**, empty, empty, **black**, empty],

[empty, empty, **black**, **white**, empty, empty],

[empty, empty, **white**, empty, empty, empty],

[empty, empty, empty, empty, empty, empty,

[empty, empty, empty, empty, empty, empty]

]).

**Situação Final:**

finalBoard([

[empty, empty, empty, black, empty, empty],

[empty, empty, white, black, black, empty],

[empty, empty, black,white, **white**, empty],

[black, white, white, **white**, black, empty],

[empty, empty, **white**, white, empty, empty],

[empty, **white**, empty, black, empty, empty]

]).

Nota: Este é apenas um exemplo de uma situação final observada num tabuleiro 6x6. Apesar disto, o tabuleiro não tem um limite definido. Apenas será definido um limite devido a questões de organização e representação (20x20 para que haja possibilidade de várias jogadas).

**Visualização do Tabuleiro**

Demonstração do código que será utilizado para mostrar o tabuleiro na consola.

**Símbolos:**

symbol(empty, S) :- S=’ ‘.

symbol(white, S) :- S=’W’.

Symbol(black, S) :- S=’B’.

**Tabuleiro:**

printBoard([Head|Tail]):-

write(‘|’),

printLine(Head),

nl,

printBoard(Tail).

printBoard([]);

**Linha:**

printLine([Head|Tail]):-

symbol(Head,S),

write(S),

write(‘|’),

printLine(Tail).

printLine([]).

**Movimentos:**