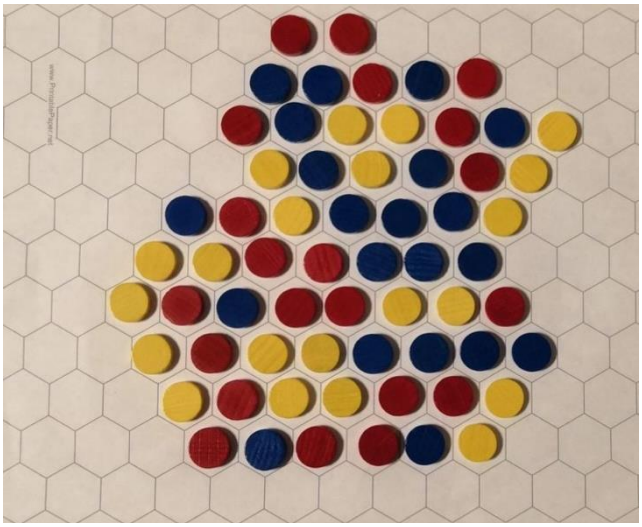
U.PORTO
FEUP ENGENHARIA

PLOG 2019/2020 – Trabalho 1

Asterismo

Relatório Intercalar

(20 de outubro de 2019)



José António Barbosa Fonseca Guerra **up201706421**@fe.up.pt

Martim Pinto da Silva **up201705205**@fe.up.pt

Índice

Asterismo

Este jogo de tabuleiro foi criado em 2019 por Giuliano Polverari.

É um jogo cooperativo para ser jogado entre 2 pessoas jogado num tabuleiro com peças hexagonais.

Há 63 peças de 3 cores diferentes (vermelho, amarelo e azul).

O objetivo de cada jogador é obter 5 peças de cada cor, sem perturbar o equilíbrio da arvore de jogo.

É um jogo cooperativo, no qual os 2 jogadores jogam em equipa por um objetivo em comum: *acumular cada um 5 peças de cada cor.*

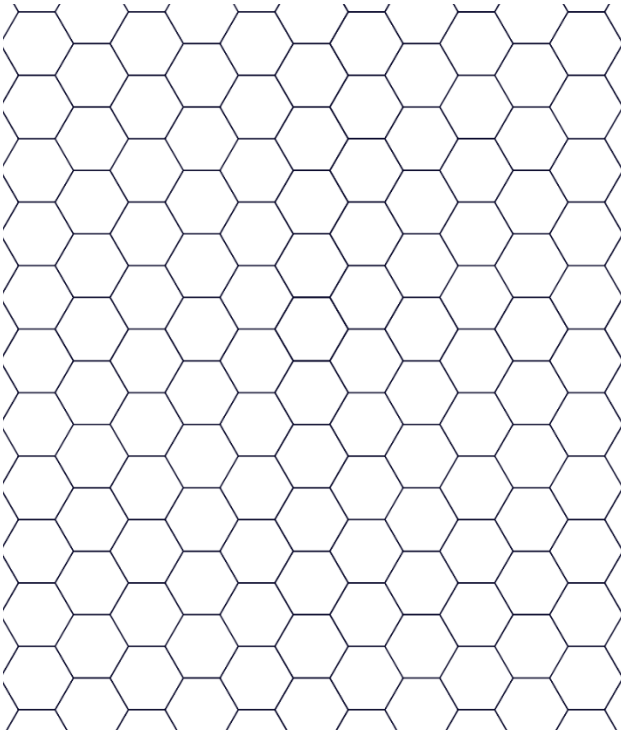


Figura 1: tabuleiro do jogo

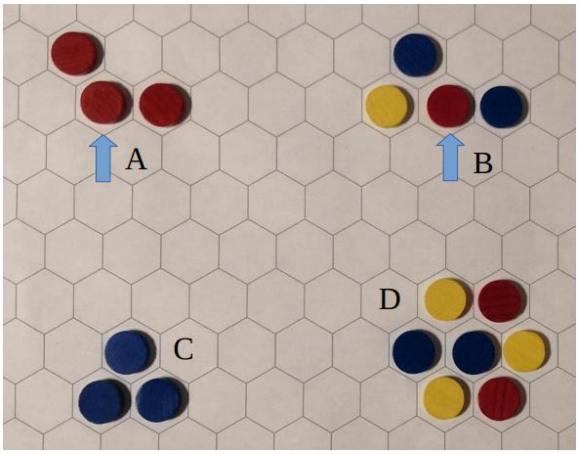
Prémios:

- 2º lugar Cogita 2019 ("melhor jogo abstrato de

Regras:

Qualquer peça pode ser retirada, no entanto as peças adjacentes têm de estar seguras.

Uma peça está segura se estiver ligada a duas peças da mesma cor (situação A) ou ligada a 3 peças de qualquer cor (situação B).

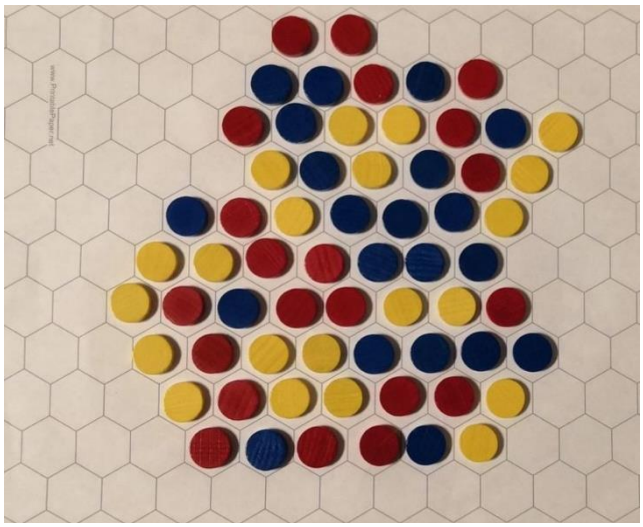


Cada jogador por turno faz apenas uma jogada, ou seja remove 1 peça, se possível.

Cada jogador tem um limite máximo de 5 peças para cada cor.

Ao retirarem peças, os jogadores têm de verificar se não quebram a arvore.

Se os 2 jogadores chegarem as 5 peças de cada cor vencem “o jogo”, senão “o jogo” vence e os jogadores saem derrotados.



Situação inicial:

As peças são colocadas aleatoriamente pelo tabuleiro.

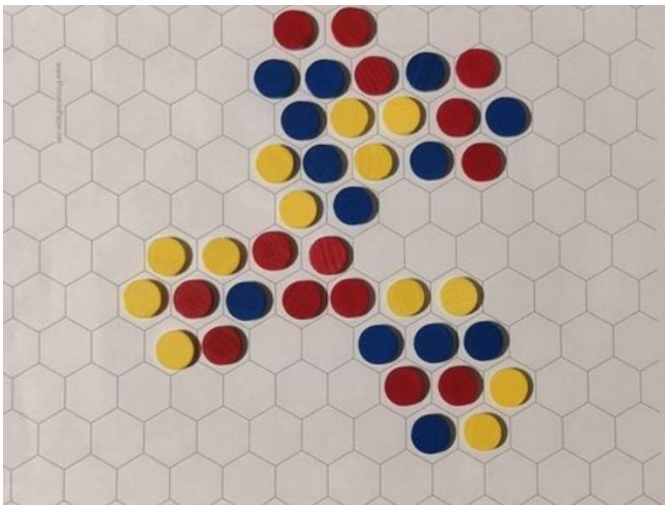
As peças, que devido a configuração inicial não estejam seguras, terão de ser atribuídas a cada jogador da forma aleatória em igual número.

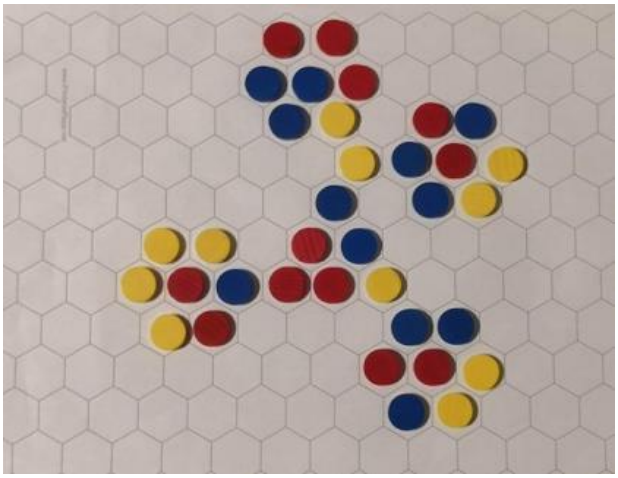
Desenvolvimento:

Vai-se retirando e colocando peças, respeitando as regras de jogo, tendo como objetivo principal acumular 5 peças de cada cor.

Os 2 jogadores jogam cooperativamente, ou seja, têm de acumular os dois 5 peças de cada cor para alcançar a vitória.

O jogador terá de ter especial atenção para não quebrar a arvore do jogo, isto é não poderá haver blocos sem peças adjacentes, ….





Situação final:

O jogador terá de estar atento a situações em que possui 14 peças, assim como o seu colega de equipa.

Nestas situações, é provável que não seja possível retirar mais peças sem quebrar a árvore do jogo. Nesta condição perdem os jogadores e ganha “o jogo”, tal como neste exemplo. Se por outro lado for possível cada 1 alcançar as 5 peças de cada cor os 2 jogadores vencem “o jogo”.

Representação interno do estado do jogo

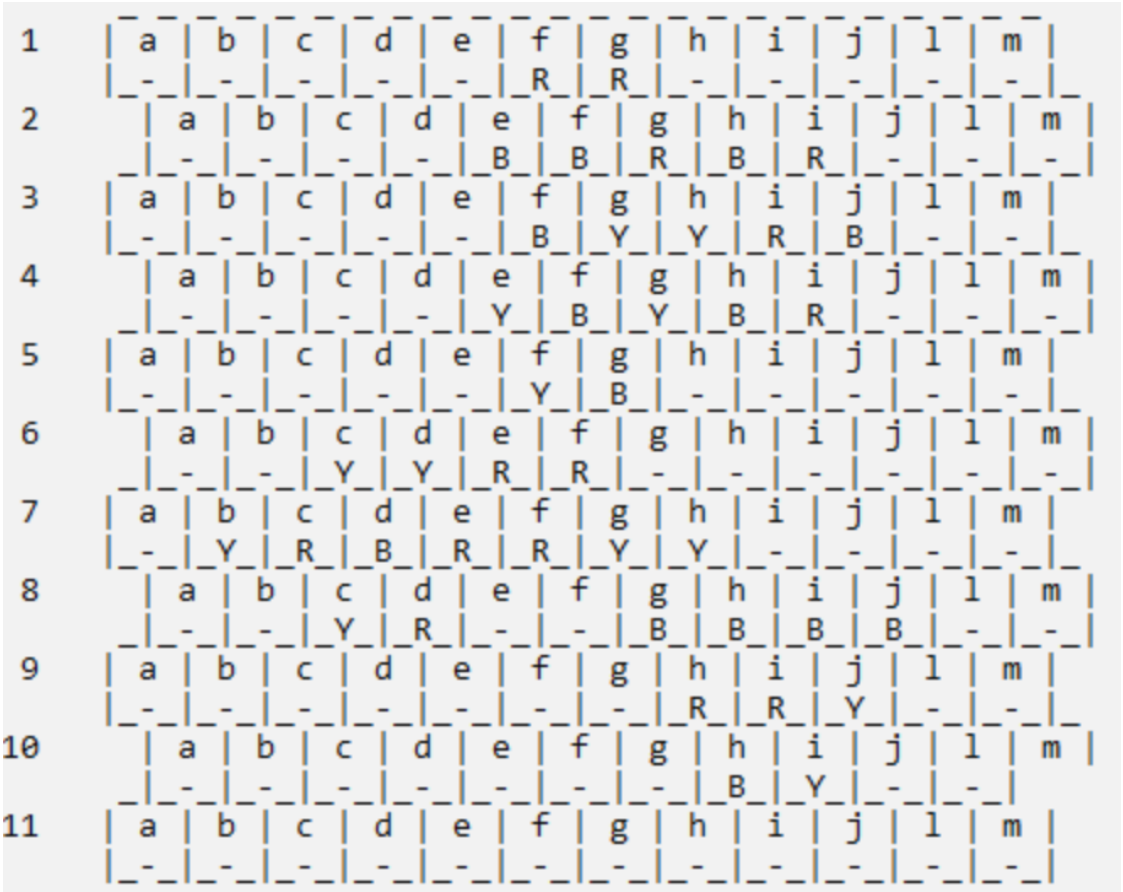
Situação inicial:

```
initialBoard([
    [null, null, null, null, null, red, red, null, null, null, null, null],
    [null, null, null, null, blue, blue, red, blue, red, null, null, null],
    [null, null, null, null, red, blue, yellow, yellow, red, blue, null, null],
    [null, null, null, null, yellow, blue, yellow, blue, red, yellow, null, null],
    [null, null, null, blue, red, yellow , blue, blue , blue , yellow , null, null],
    [null, null, yellow, yellow, red, red, blue, blue, blue, null, null, null],
    [null, yellow, red, blue, red, red, yellow, yellow, red, null, null, null],
    [null, null, yellow, red, yellow, yellow, blue, blue, blue, blue, null, null],
    [null, null, null, yellow, red, yellow, yellow, red, red, yellow, null, null],
    [null, null, null, null, red, blue, red, red, blue, yellow, null, null],
    [null, null, null, null, null, null, null, null, null, null, null, null]]
).
```

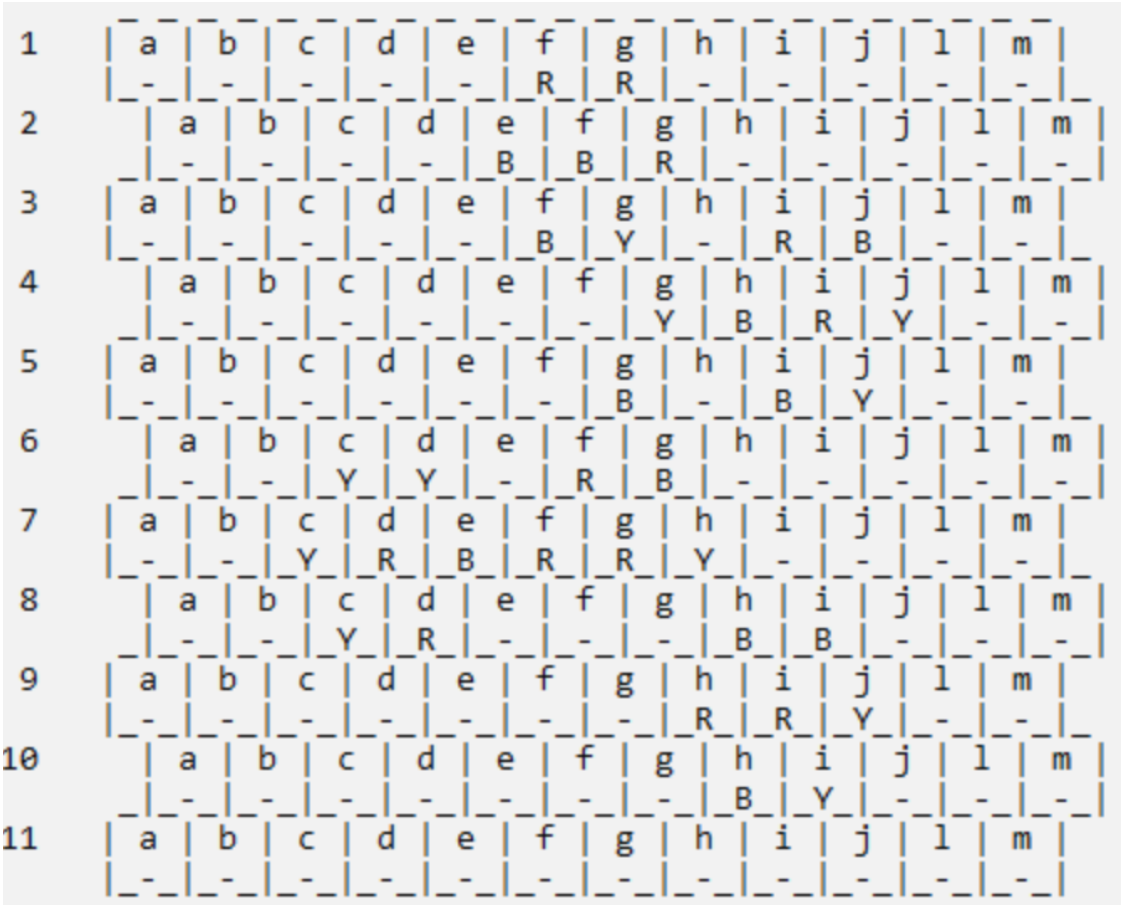
```
    _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
1   | a | b | c | d | e | f | g | h | i | j | l | m |
    |_-_|_-_|_-_|_-_|_-_|_R_|_R_|_-_|_-_|_-_|_-_|_-_|
2     | a | b | c | d | e | f | g | h | i | j | l | m |
      |_-_|_-_|_-_|_-_|_B_|_B_|_R_|_B_|_R_|_-_|_-_|_-_|
3   | a | b | c | d | e | f | g | h | i | j | l | m |
    |_-_|_-_|_-_|_-_|_R_|_B_|_Y_|_Y_|_R_|_B_|_-_|_-_|_
4     | a | b | c | d | e | f | g | h | i | j | l | m |
      |_-_|_-_|_-_|_-_|_Y_|_B_|_Y_|_B_|_R_|_Y_|_-_|_-_|
5   | a | b | c | d | e | f | g | h | i | j | l | m |
    |_-_|_-_|_-_|_B_|_R_|_Y_|_B_|_B_|_B_|_Y_|_-_|_-_|_
6     | a | b | c | d | e | f | g | h | i | j | l | m |
      |_-_|_-_|_Y_|_Y_|_R_|_R_|_B_|_B_|_B_|_-_|_-_|_-_|
7   | a | b | c | d | e | f | g | h | i | j | l | m |
    |_-_|_Y_|_R_|_B_|_R_|_R_|_Y_|_Y_|_R_|_-_|_-_|_-_|_
8     | a | b | c | d | e | f | g | h | i | j | l | m |
      |_-_|_-_|_Y_|_R_|_Y_|_Y_|_B_|_B_|_B_|_B_|_-_|_-_|
9   | a | b | c | d | e | f | g | h | i | j | l | m |
    |_-_|_-_|_-_|_Y_|_R_|_Y_|_Y_|_R_|_R_|_Y_|_-_|_-_|_
10    | a | b | c | d | e | f | g | h | i | j | l | m |
      |_-_|_-_|_-_|_-_|_R_|_B_|_R_|_R_|_B_|_Y_|_-_|_-_|
11  | a | b | c | d | e | f | g | h | i | j | l | m |
    |_-_|_-_|_-_|_-_|_-_|_-_|_-_|_-_|_-_|_-_|_-_|_-_|
```

Desenvolvimento:

```
initialBoard([
        [null, null, null, null, null, red, red, null, null, null, null, null],
        [null, null, null, null, blue, blue, red, blue, red, null, null, null],
        [null, null, null, null, null, blue, yellow, yellow, red, blue, null, null],
        [null, null, null, null, yellow, blue, yellow, blue, red, null, null, null],
        [null, null, null, null, null, yellow , blue, null, null,null, null, null],
        [null, null, yellow, yellow, red, red, null, null, null, null, null, null],
        [null, yellow, red, blue, red, red, yellow, yellow, null, null, null, null],
        [null, null, yellow, red, null, null, blue, blue, blue, blue, null, null],
        [null, null, null, null, null, null, null, red, red, yellow, null, null],
        [null, null, null, null, null, null, null, blue , yellow, null, null],
        [null, null, null, null, null, null, null, null, null, null, null, null]]
    ).
```



Situação final:




```
initialBoard([
        [null, null, null, null, null, red, red, null, null, null, null, null],
        [null, null, null, null, blue, blue, red, null, null, null, null, null],
        [null, null, null, null, null, blue, yellow, null, red,  blue, null, null],
        [null, null, null, null, null, null, yellow, blue, red, yellow, null, null],
        [null, null, null, null, null, null, blue, null, blue, yellow, null, null],
        [null, null, yellow, yellow, null, red, blue, null, null, null, null, null],
        [null, null, yellow, red, blue, red, red, yellow, null, null, null, null],
        [null, null, yellow, red, null, null, null, blue, blue, null, null, null],
        [null, null, null, null, null, null, null, red, red, yellow, null, null],
        [null, null, null, null, null, null, null, blue , yellow, null, null],
        [null, null, null, null, null, null, null, null, null, null, null, null]]
    ).
```

Visualização do tabuleiro

```
printLineConst :-
    write('\n---|---|---|---|---|---|---|---|---|---|---|---|---|\n').

printBoardTop :-
    write('      _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _\n').


printBoardUp :-
    write('| a | b | c | d | e | f | g | h | i | j | l | m |\n ').


printBoardLine([], Line) :-
    (
        Line==12, write('|\n');
        Aux is Line mod 2, Aux==0, write('|_\n');
        write('|\n')
    ).

printBoardLine([H|T], Line) :-
    write('|_'), piece(H, S),

    ansi_format([bold,fg(cyan)], '~c', [S]),
    %write(S),
    write('_'),
    printBoardLine(T, Line).

printBoardBody([], 12).
printBoardBody([H|T], Line) :-
    (
        Line<10, write(' ');
        Line>9
    ),

    write(Line), write('   '),
    Mod is Line mod 2,
    (
        Mod==0, write('  ') ;
        Mod==1
    ),
    printBoardUp,
    write('    '),
    (
        Mod==0, write(' _');
        Mod==1
    ),
    LineI is Line+1,
    printBoardLine(H, LineI),
    printBoardBody(T, LineI).


printBoard(X) :-
    printBoardTop,
    printBoardBody(X, 1).

print :-
    initialBoard(Init),
    printBoard(Init).
```

Predicados

Cabeçalho do predicado para adicionar e remover uma peça (remover no sentido de colocar um ‘-‘ no espaço que ficou vazio)

```
addPiece(InitialBoard, Color, Row, Column, NewBoard).
```