U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Programação em Lógica
EIC0026

Ano Letivo 2015/2016

Relatório Intercalar

"DUPLOHEX"

Grupo DuploHex_3

Carlos Manuel Carvalho Boavista Samouco
up201305187@fe.up.pt

Diogo Belarmino Coelho Marques
up201305642@fe.up.pt

11 de Outubro de 2015

1. O Jogo DuploHex

➔ **O que é?** [1] [2]

Trata-se de um jogo de estratégia para dois jogadores criado por José Manuel Astilleros García-Monge, que teve como base da sua criação o *Hex*, jogo de tabuleiro criado em 1942 por Piet-Hein jogado tradicionalmente num tabuleiro em forma de losango com dimensões 11 x 11 onde o objetivo era conectar os lados opostos do tabuleiro com peças da mesma cor.

Ao contrário do seu "antecessor," o *Duplohex* introduz dois tipos diferentes de peças (discos e anéis), em vez da peça única. Esta variação no tipo e número de peças a que cada jogador tem direito vem proporcionar um novo leque de estratégias e possibilidades que não existiam no seu antecessor.

À semelhança do Hex, o objetivo de cada jogador é conectar os lados opostos do tabuleiro marcados pela sua cor com o mesmo tipo de peça, utilizando tanto discos como anéis.

➔ **Constituição** [3]

- Tabuleiro de *hex* 7 x 7
- 24 discos brancos
- 24 discos pretos
- 24 anéis brancos
- 24 anéis pretos

[1] https://pt.wikipedia.org/wiki/Hex_(jogo)
[2] https://boardgamegeek.com/boardgame/174474/duplohex
[3] http://www.nestorgames.com/#duplohex_detail

➔ **Regras** [4]

O tabuleiro encontra-se inicialmente vazio. Cada jogador deve escolher a sua cor (preto ou branco). O jogador de cor branca começa o jogo por colocar uma das suas peças (um disco ou um anel) em qualquer célula do tabuleiro (neste estado encontram-se todas livres).

Nas jogadas seguintes, começando com a cor preta, cada jogador deve, à vez, realizar dois movimentos obrigatórios: um movimento com um disco e um movimento com um anel da sua cor segundo uma ordem arbitrária.

Possiveis movimentos que um jogador pode realizar com:

- ...um disco:
    - colocar um disco numa casa vazia
    - mover um dos seus discos previamente colocados no tabuleiro para dentro de um anel qualquer que esteja colocado numa casa vizinha
- … um anel:
    - colocar um anel nunca casa vazia
    - mover um dos seus anéis previamente colocados no tabuleiro para um disco qualquer que esteja colocado numa casa vizinha

Assim que um disco é colocado dentro de um anel ou que um anel cobre um disco, nenhum dos dois pode ser mais movido durante o jogo.

Os jogadores não podem passar a sua vez. As peças não podem ser empilhadas na mesma casa (apenas é permitido um par anel-disco em cada uma). Por último, no caso de um jogador não puder fazer a sua jogada, ele deve na sua vez adicionar um dos seus anéis ou discos numa célula qualquer do tabuleiro ocupada por um disco ou anel, respetivamente.

O jogo termina no momento em que um dos jogadores consegue fazer uma sequência com discos ou anéis da sua cor (mas nunca uma combinação de ambos) conectando os dois lados opostos do tabuleiro.

Este jogo também pode ser jogado num tabuleiro com dimensões 6 x 6, de modo a tornar o jogo menos extenso. Nesta configuração, e recorrendo a um tabuleiro de 7 x 7, preenche-se as duas paredes de cada jogador com discos e anéis da cor correspondente antes de iniciar o jogo.

[4] http://www.nestorgames.com/rulebooks/DUPLOHEX_EN.pdf

2 Representação do estado do jogo

2.1 Visualização do tabuleiro

Tendo em conta as limitações do *SICStus Prolog*, na impossibilidade de representar o tabuleiro e as peças com cores diferentes, foi elaborada a seguinte correspondência entre letras e peças*:*

‘ d ’ : disco branco
‘ D ’ : disco preto

‘ a ’ : anel branco
‘ A ’ : anel preto

‘ B ’ : anel branco em disco branco
‘ b ’ : anel branco em disco preto

‘ P ’ : anel preto em disco preto
‘ p ’ : anel preto em disco branco

‘ \ ’ : parede do jogador da cor branca
‘ - ’ : parede do jogador da cor preta

Para peças simples, a letra representa a inicial da peça (**d :** disco, **a** : anel). Se estiver em minúsculas, representa uma peça de cor branca, caso contrário representa uma peça de cor preta.

Para pares de peças, a letra representa a cor do anel emparelhado com um disco. Se estiver em maiúsculas, é porque esse anel está emparelhado com um disco da mesma cor, caso contrário, esse anel está emparelhado com um disco de cor diferente.

2.2 Representação interna do estado

Para a representação interna do estado do jogo, escolheu-se uma variável composta, constituída por quatro variáveis simples, separadas por hífenes.

- NextTurn
    - átomo que representa a próxima cor a ser jogada (*black* ou *white*).
- BlackPieces
    - lista com a seguinte estrutrura [*discos*, *aneis*]. **Ex.:** [5, 1] significa que o jogador de cor preta tem 5 discos e 1 anel disponíveis para jogar no tabuleiro.
- WhitePieces
    - lista com a seguinte estrutura [*discos*, *aneis*]. **Ex.:** [2, 6] significa que o jogador de cor branca tem 2 discos e 6 anéis disponíveis para jogar no tabuleiro.
- Board
    - internamente uma lista de listas: **[** [_,_,_,...], [_,_,_,...], ...**]**, que traduz uma representação em 2D do tabuleiro onde cada lista 'interna' corresponde a uma linha, e os elementos das diferentes listas formam as diferentes colunas. O tabuleiro terá dimensões do tipo (X,Y) com X = Y, isto é, o número de elementos em cada linha do tabuleiro deve ser igual ao número total de linhas.

3. Visualização do tabuleiro

```
print_all(Board-NextTurn-Player1-Player2) :-
        write('----------------------------'),
        print_table(Board, 0),
        nl,
        write('       ----------------------------').
```

O predicado **print_all/1** recebe uma estrutura de dados, na qual uma das variáveis é uma lista de listas (contendo a representação interna do tabuleiro do jogo) e imprime no ecrã uma visualização completa do respectivo tabuleiro, já com as paredes horizontais. Estas paredes são representadas por hífens **‘ - ’**.

```
print_char(_, 0).
print_char(X, Y) :- Y > 0, write(X), Y1 is Y - 1, print_char(X, Y1).
```

O predicado **print_char/2** recebe um literal caracter e um literal inteiro, e tal como o nome indica permite imprimir um caracter X no ecrã Y vezes. Por enquanto é apenas usado para imprimir espaços no tabuleiro, permitindo a sua visualização na diagonal, à semelhança do tabuleiro verdadeiro.

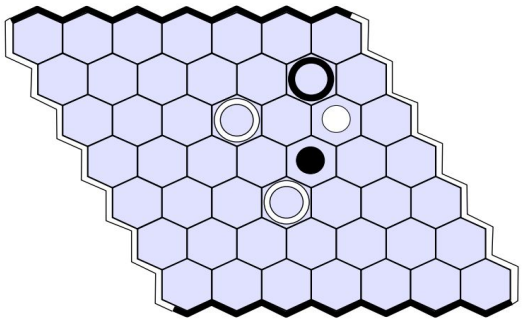
```
print_table([], _).
print_table([H|T], X) :-
        nl,
        print_char(' ', X),
        write(' \\ '),
        print_line(H),
        X1 is X + 1,
        print_table(T, X1).
```

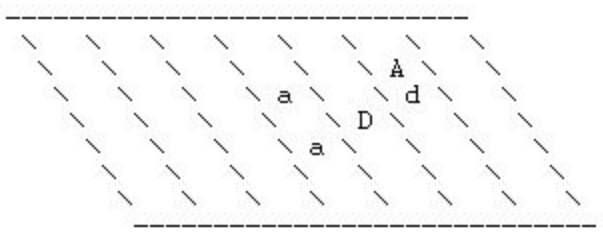
O predicado **print_table/2** recebe uma lista de listas (a representação interna do tabuleiro) e um literal inteiro inicialmente a 0 (representando o índice da linha atual), imprimindo o respetivo tabuleiro no ecrã, embora incompleto (apenas as células e as paredes laterais). Este procedimento é complementado pelo **print_all/1**, que imprime também as paredes horizontais.

```
print_line([]).
print_line([H|T]) :- write(H), write(' \\ '), print_line(T).
```

O predicado **print_line/1** recebe uma lista (a representação interna de uma linha do tabuleiro) e imprime recursivamente no ecrã todas as células dessa linha separadas por uma barra **‘ \ ’**.

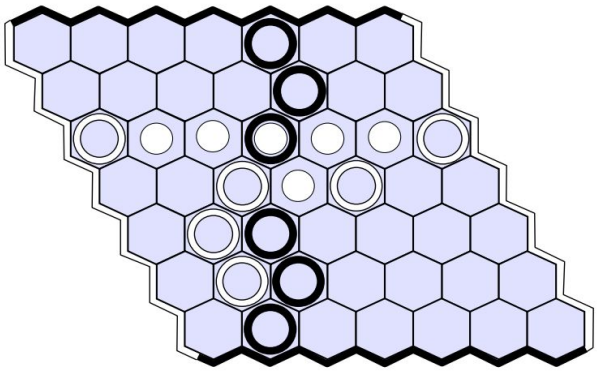
- estado inicial

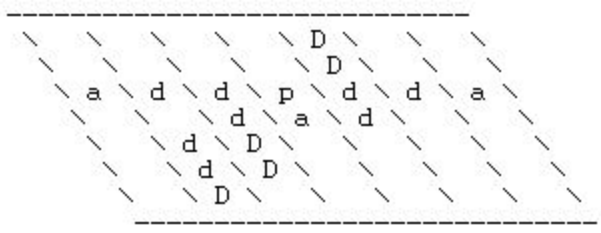




print_all([[' ', ' ', ' ', ' ', ' ', ' ', ' '],
[' ', ' ', ' ', ' ', ' ', 'A', ' '],
[' ', ' ', ' ', 'a', ' ', 'd', ' '],
[' ', ' ', ' ', ' ', 'D', ' ', ' '],
[' ', ' ', ' ', 'a', ' ', ' ', ' '],
[' ', ' ', ' ', ' ', ' ', ' ', ' '],
[' ', ' ', ' ', ' ', ' ', ' ', ' ']]-black**-player(**white, 23, 23**)-player(**black, 23, 22**)).**

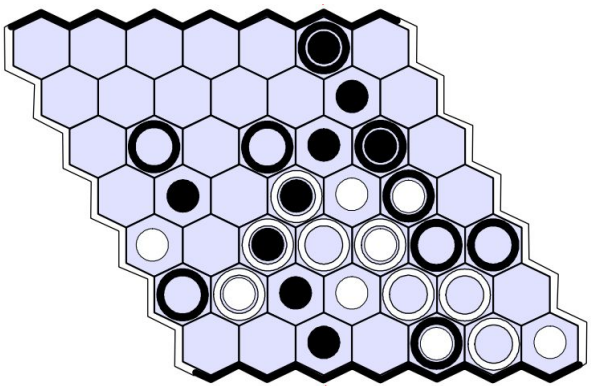
- estado intermédio

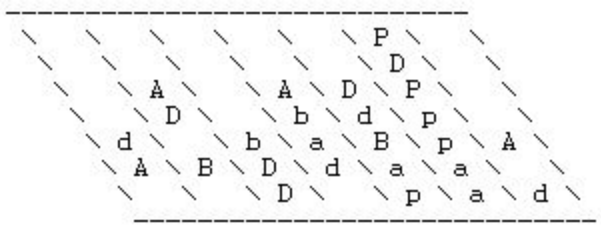




print_all([[' ', ' ', ' ', ' ', 'D', ' ', ' '**],**
[' ', ' ', ' ', ' ', 'D', ' ', ' '**],**
[a, d, d, p, d, d, a**],**
[' ', ' ', d, a, d, ' ', ' '**],**
[' ', d, 'D', ' ', ' ', ' ', ' '**],**
[' ', d, 'D', ' ', ' ', ' ', ' '**],**
[' ', 'D', ' ', ' ', '', ' ', ' '**]]**-black**-player(**white, 24, 18**)-player(**black, 18, 18**)).**

- estado final (vitória do jogador de cor preta)





print_all([[' ', ' ', ' ', ' ', ' ', 'P', ' '**],**
[' ', ' ', ' ', ' ', ' ', 'D', ' '**],**
[' ', 'A', ' ', 'A', 'D', 'P', ' '**],**
[' ', 'D', ' ', b, d, p, ' '**],**
[d, ' ', b, a, 'B', p, 'A'**],**
['A', 'B', 'D', d, a, a, ' '**],**
[' ', ' ', 'D', ' ', p, a, d**]]-**void**-player(**white, 15, 15**)-player(**black, 16, 16**)).**

(representação de vários estados do jogo, bem como a sua visualização em Prolog)

4. Movimentos

4.1 Movimentos que um jogador pode realizar com um disco

➔ **placeDisc(**Board, **player(**Player,Discs, Rings**)**, X, Y**)**
colocar um dos seus discos numa célula vazia do tabuleiro *Board* com coordenadas (X, Y) - esta célula não pode estar ocupada por um outro disco ou anel, nem por ambos

➔ **moveDiscToRing(**Board, **player(**Player, Discs, Rings**)**, fromX, fromY, toX, toY**)**
mover um dos seus discos já colocados no tabuleiro *Board* de uma célula com coordenadas (fromX, fromY) para dentro de um anel (branco ou preto) localizado numa célula vizinha com coordenadas (toX, toY) - a célula de destino não pode estar vazia nem emparelhada (ocupada por um anel e por um disco)

Um disco que estiver dentro de um anel não pode ser mais movido durante o jogo.

4.2 Movimentos que um jogador pode realizar com um anel

➔ **placeRing(**Board, **player(**Player, Discs, Rings**)**, X, Y**)**
colocar um dos seus anéis numa célula vazia do tabuleiro *Board* com coordenadas (X,Y) - esta célula não pode estar ocupada por um outro disco ou anel, nem por ambos

➔ **moveRingToDisc(**Board, **player(**Player, Discs, Rings**)**, fromX, fromY, toX, toY**)**
mover um dos seus anéis já colocados no tabuleiro *Board* de uma célula com coordenadas (fromX, fromY) para uma célula vizinha com coordenadas (toX, toY) que tenha sido ocupada por um disco (branco ou preto) - a célula de destino não pode estar vazia nem emparelhada (ocupada por um anel e por um disco)

Um anel com um disco no seu interior não pode ser mais movido durante o jogo.