Zurero

Relatório Intercalar

Mestrado Integrado em Engenharia Informática e Computação

Programação em lógica

Grupo: Zurero_5

Elementos do grupo:

Ana Isabel Ferreira Maia - 201504108

André Filipe de Soveral Torres Lopes dos Santos - 200505634

Outubro 2018

Índice

1.Introdução

Este trabalho foi realizado no âmbito da unidade curricular de Programação em Lógica, primeiro semestre do segundo ano do Mestrado Integrado em Engenharia Informática e Computação.

O nosso grupo escolheu o jogo Zurero por ser um jogo interessante com regras diferentes ao que se vê na maior parte dos outros jogos, devido ao movimento das peças durante o jogo. O objetivo deste trabalho era implementar o jogo Zurero na linguagem Prolog com quatro modos de jogo: Humano vs Humano, Computador vs Computador, Humano vs Computador e Computador vs Humano. Estas últimas duas têm uma diferenciação muito clara pelo facto que o primeiro jogador é o único que pode colocar a sua peça livremente no tabuleiro.

2. Zurero

2.1 Introdução

Zurero é um jogo de tabuleiro em que dois jogadores tentam colocar 5 peças da mesma cor numa linha (horizontal, vertical ou diagonal).

É jogado num tabuleiro quadrado, tradicionalmente com 19x19 interseções (o mesmo usado no jogo Go), mas outros tamanhos também podem ser usados, onde se podem colocar as peças pretas ou brancas.

O objetivo deste jogo é simples, colocar 5 peças da mesma cor numa linha (horizontal, vertical ou diagonal). O primeiro jogador a alcançar este objetivo ganha.

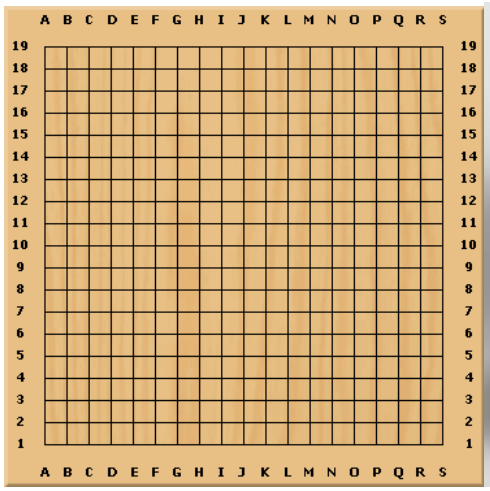

Imagem 1-O tabuleiro de Zurero tradicional. Imagem 2-Tabuleiro Go

2.2 História

Zurero foi um jogo inventado por Jordan Goldstein em 2009. É uma variação interessante do jogo “Go-Moku” (em japonês), em que a grande diferença está quando a peça deslizante encontra uma peça com com um espaço vazio atrás dela. No “Go-Moku” a peça não se mexe, porém no Zurero, como foi visto anteriormente, a peça move-se uma interseção para a frente e a peça deslizante ocupa então a anterior posição da peça.

2.3 Regras

O jogo começa com um tabuleiro vazio.

Cada jogador tem uma cor alocada, geralmente preto ou branco. O jogador com as peças pretas começa colocando uma pedra preta no meio do tabuleiro.

Em seguida, os jogadores irão, à vez, deslizar uma pedra da sua cor alocada pelo tabuleiro, começando em qualquer uma das bordas e deslizando ao longo de uma linha, certificando-se que existe pelo menos uma pedra nessa linha.

A pedra desliza até bater na primeira pedra que encontrar, se esta pedra não tiver nada atrás dela, é "empurrada" um passo para trás e a pedra deslizante move para a interseção que a pedra anterior ocupava.

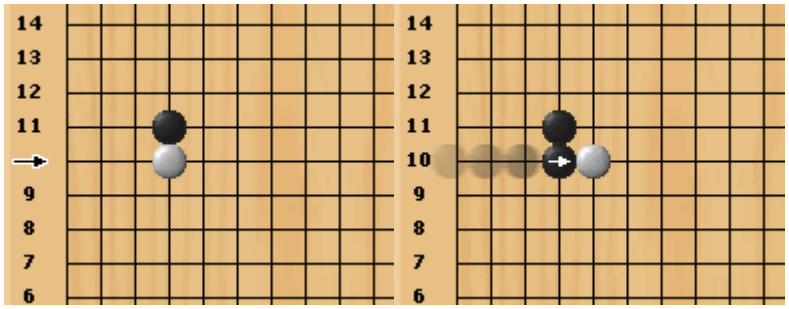

Imagem 3-Antes do movimento *Imagem 4-Após o movimento*

Por outro lado, se a pedra deslizante bate contra uma pedra que tem outra "atrás" desta, nenhuma delas se move, e a peça deslizante fica uma interseção "à frente da peça em que embateu.

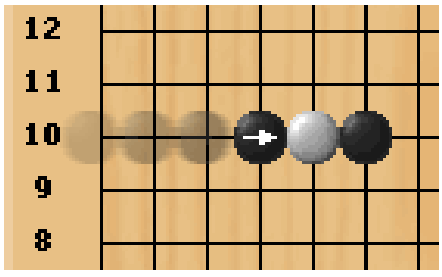

Imagem 5-Jogada sem deslizamento das peças já existentes

3. Lógica do Jogo

3.1 Representação Interna do Estado de Jogo

Neste Jogo será utilizado uma lista de listas para representar o tabuleiro de jogo internamente. Esta lista será composta por 19 listas, sendo estas mesmas compostas por 19 elementos. As casas vazias serão representadas por um “0”, as peças pretas por “1” e finalmente as peças brancas por “2”.

Este jogo tem uma particularidade, nomeadamente em relação à primeira jogada, uma vez que a primeira peça poderá ser colocada em qualquer lado do tabuleiro. As restantes jogadas são realizadas “à volta” das peças já presentes, logo essa primeira jogada terá uma implementação diferente das demais.

As jogadas que não a primeira, serão sempre à volta das peças já presentes e no fim de cada jogada (e tabuleiro resultante), verificar-se-á se alguma das equipas cumpre os requisitos de vitória (dando sempre prioridade ao jogador que fez a última jogada em caso de vitória “simultânea”)

```
tabInicial = [[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],
              [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
              [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],
              [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
              [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],
              [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
              [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],
              [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
              [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
```

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]].

Tabuleiro a meio do jogo

```
tabactual = [[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],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [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],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,2,1,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,1,1,2,0,0,0,0,0,0,0,0],
             [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],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [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],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]].
```

Tabuleiro no fim de um jogo (vitória das brancas):

```
tabactual = [[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],
             [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
             [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],
             [0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0],
```

[0,0,0,0,0,2,2,2,2,2,1,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,1,2,1,1,2,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,1,0,0,2,0,0,0,0,0,0,0,0],

[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],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[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],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]].

3.2 Visualização do Tabuleiro em modo de texto

Para a visualização do tabuleiro foram usados os seguintes predicados:

- display_game(+B, +P) - predicado de visualização do tabuleiro
- abc(+N) - faz display das letras necessárias para melhor ler as colunas do tabuleiro
- print_tab(+B, +N) - predicado que inicia a recursão de apresentar o tabuleiro, e ainda faz display dos números de cada linha para melhor ler o tabuleiro
- print_line(+L) - predicado que imprime a linha do tabuleiro dada em L
- Traduz(+N, -S) - dá o símbolo a que corresponde uma peça branca, preta ou espaço vazio

Tabuleiro vazio

```
Current Player: Jogador
   A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|
19|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
18|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
17|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
16|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
15|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
14|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
13|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
12|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
11|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
10|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
9 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
8 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
7 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
6 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
5 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
4 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
3 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
2 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
1 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
yes
```

Tabuleiro a meio do jogo:

```
Current Player: Jogador
   A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|
19|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
18|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
17|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
16|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
15|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
14|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
13|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
12|+-+-+-+-+-+-+-+-+-W-B-+-+-+-+-+-+-+-+-+-+-+
11|+-+-+-+-+-+-+-+-+-B-B-W-+-+-+-+-+-+-+-+-+-+
10|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
9 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
8 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
7 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
6 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
5 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
4 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
3 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
2 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
1 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
yes
```

Tabuleiro no fim do jogo:

```
Current Player: Jogador
   A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|
19|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
18|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
17|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
16|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
15|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
14|+-+-+-+-+-+-+-+-+-B-+-+-+-+-+-+-+-+-+-
13|+-+-+-+-+-+-+-+-+-B-B-+-+-+-+-+-+-+-+-
12|+-+-+-+-+-W-W-W-W-W-B-+-+-+-+-+-+-+-+-
11|+-+-+-+-+-+-B-W-B-B-W-+-+-+-+-+-+-+-+-
10|+-+-+-+-+-+-+-B-+-+-W-+-+-+-+-+-+-+-+-
9 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
8 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
7 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
6 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
5 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
4 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
3 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
2 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
1 |+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
yes
```

3.3 Lista de Jogadas Válidas

Valid_moves(+Board, -Orientacao, -Valores) :- predicado que recebe um tabuleiro e retorna duas lista, orientação que contém a orientação(up, down, right, left) de cada movimento válido, e valores que retorna o valor ou letra dos movimentos válidos. Os elementos numa lista correspondem ao da outra, ou seja, por exemplo, o primeiro elemento de orientação é a orientação correspondente ao primeiro valor de valores. Predicado irá fazer append de todos os valid moves ditados pelo validmoveright.

Para este predicado vamos precisar dos predicados: **turnValueToLetter**(+L, -L1), **validmoveright**(+Board, +It, +O, -Orientacao, -Valores), **confere**(+L), **tempeca**(+L), **naoocupado**(+L), **twopieces**(+L), **inverterBoardH**(+BI, -BF), **transpose,** **turnValuetoLetter**(+Linha, -LinhaF)**.**

3.4 Execução de Jogadas

Move (+Jogador, +BoardInicial, +Orientacao, +Iterador, -BoardFinal) -predicado que vai fazer um movimento na direção e linha ou coluna desejada. Iterador, caso a orientação seja up or down, será Letra-A, convertido em código ascii, caso seja right ou left, o iterador será 19-Número da linha

Para este predicado vão ser precisos os predicados: **inverterBoardH**(+BI, -BF), **inverterBoardV**(+BI, -BF), **inverterLinha**(+Lista, -InvLista), **moveLeft**(+Jogador, +BoardInicial, +Iterador, -BoardFinal), **placeT**(+Jogador, +BoardInicial, +Iterador, -BoardFinal), **placeF**(+Jogador, +BoardInicial, +Iterador, -BoardFinal), **find**(+Linha, +V, -Res), **findEmpurrar**(+Linha, +Res, -Jog), **check**(+BoardInicial, -Res), **nextElementFull**(+Linha) e **linha**(+Bi, +I, -Linha), **place**(+Jogador, +BI, +Letra, +Numero, -BoardFinal).

3.5 Final do Jogo

Game_over(+Board, -Winner) - predicado que irá retornar quem ganhou, ou fail se ninguém ganhou.

Para este predicado vão ser necessários os predicados: **gameOverV**(+Board, -Winner), **gameOverH**(+Board, -Winner), **gameOverLineCheck**(+Board,+N,+Temp, -Winner), **ignorezeros**(+Linha, -LinhaRes).

3.6 Avaliação do Tabuleiro

Value(+Board, +Player, +Direcao, +It, -Value) - com este predicado conseguimos avaliar um certo movimento se é mais ou menos vantajoso. O It será, tal como no move, 19-numero ou Letra-A(transformado em código ascii). No nosso caso, o predicado irá avaliar quantas peças do Player seguidas encontramos no movimento desejado e retornamos esse mesmo número de peças seguidas.

Para o funcionamento deste predicado iremos usar: **ignorezeros**(+Linha, -Res), **countline**(+Linha, +Player, +Temp, +Value) e outras funçoes já definidas anteriormente como inverterBoardH e linha.

3.7 Jogada do Computador

Choose_move(+Board, +Player, +Dificuldade, -Move) - predicado que vai escolher, conforme o nível de dificuldade e o Jogador, uma certa jogada para o computador.

Este predicado irá então dividir entre fácil e difícil em termos de dificuldade. Em fácil, usando os **valid_moves**, escolhemos um movimento válido "à sorte" com o uso do predicado **random**. Em difícil, usamos o predicado **bestmove**(+Board, +Jogador, +Orientacao, +Valores, +Temp,-MOTemp,-MVTemp, -MelhorOrientacao, -MelhorValor) , que dentro dos movimentos válidos vai encontrar qual tem o maior value e devolver, e o predicado **turnLetterToIterator**(+Valor, -ValorRes). Usamos também o predicado **choose**(+Line, +R, -Res) para a dificuldade fácil.

4. Conclusões

Com este trabalho adquirimos conhecimento e familiaridade com o Prolog. Porém temos que destacar que devido a natureza do Prolog e dos seus IDE's tivemos problemas com o debugging e a sua falta de eficiência.

Dificuldades:

- Algumas funções (por exemplo o choose_move e suas funçoes auxiliares), revelaram-se mais complicadas que o esperado na sua implementação.
- Alguns erros e bugs são muito situacionais, i.e. só ocorrem em ocasiões especificas.

Poderia ser melhorado:

- Alguns bugs que se revelaram mais complicados poderiam ser melhorados, pois alguns erros se revelavam em situações muito especificas.
- Introdução de um método de input para o utilizador, que seja resistente a erros e seja mais intuitivo para o utilizador.

5. Bibliografia

http://www.iggamecenter.com/info/en/zurero.html

https://rpggeek.com/boardgame/41145/zurero

https://rpggeek.com/boardgame/11929/go-moku

https://pt.wikipedia.org/wiki/Gomoku