Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Squex

Grupo Squex_2, turma 5

Professor: Rui Carlos Camacho de Sousa Ferreira da Silva

Desenvolvido por:
David Freitas Dinis, up201706766@fe.up.pt
José Miguel Martins Gomes, up201707054@fe.up.pt

Introdução

Este trabalho tem como objetivo principal a realização de um jogo de tabuleiro para dois jogadores em linguagem Prolog. É possível caracterizar um jogo de tabuleiro pelo tipo do tabuleiro e das peças e pelas suas regras de jogo.

A aplicação a desenvolver permitirá três modos de utilização que serão Humano vs Humano, Humano vs Computador e Computador vs Computador.

O Jogo Squex

Squex é um jogo de tabuleiro de “conexão” para dois jogadores e foi adicionado ao website “boardgamegeek.com” pela primeira vez em 2019.

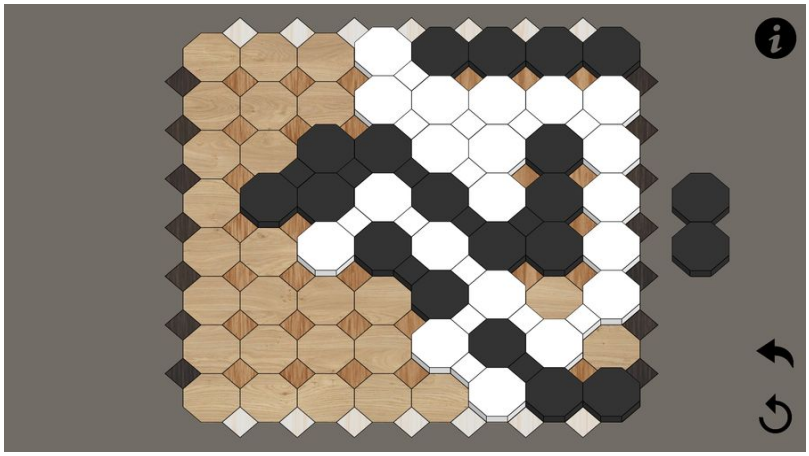

Figura 1 - Tabuleiro durante o jogo[1]

O jogo decorre num tabuleiro 8x8 composto por octógonos e quadrados e o objetivo de cada jogador é criar uma conexão entre os seus dois lados do tabuleiro com uma linha contínua de peças.

As regras do jogo são as seguintes:

1. À vez, cada jogador coloca uma peça octogonal em qualquer octágono livre no tabuleiro.
2. Se uma peça for colocada diagonalmente em relação a uma peça da mesma cor, é colocado um quadrado da mesma cor a unir as duas peças (é possível colocar até quatro quadrados com apenas uma jogada).
3. Se ocorrer uma situação da regra 2 mas já existir um quadrado do adversário naquela posição, esse quadrado é substituído por um quadrado da cor de quem jogou. Este movimento é chamado de “corte”.
4. Quando um jogador realiza um “corte” o seu adversário pode jogar duas vezes no turno seguinte.
5. Um jogador ganha apenas quando conectar os seus dois lados do tabuleiro e for impossível que o adversário quebre a conexão fazendo um “corte”.

[1] Imagem retirada do website boardgamegeek.com/boardgame/279483/squex

Modelação do Jogo em Prolog

Representação Interna

A representação interna do tabuleiro é feita através de 64 “objetos” octo e 49 “objetos” square.

Os factos octo são do tipo octo(linha, coluna, estado, id_CE, id_CD, id_BE, id_BD). Os atributos “linha” e “coluna” representam a posição do octógono no tabuleiro e o atributo “estado” representa o estado atual daquele octógono (‘@’ para o jogador A, ‘b’ para o jogador B e ‘.’ para um espaço vazio). Os atributos id_CE, id_CD, id_BE e id_BD correspondem respectivamente ao id dos quadrados posicionados relativamente ao octógono acima à esquerda, acima à direita, abaixo à esquerda e abaixo à direita (‘nn’ é utilizado nas extremidades do tabuleiro onde não existe um quadrado utilizável).

```
octo(2,0,o,nn, 7,nn,14).
octo(2,1,o, 7, 8,14,15).
octo(2,2,o, 8, 9,15,16).
octo(2,3,o, 9,10,16,17).
octo(2,4,o,10,11,17,18).
octo(2,5,o,11,12,18,19).
octo(2,6,o,12,13,19,20).
octo(2,7,o,13,nn,20,nn).
```

Figura 2 - Exemplo da representação dos octógonos

Os factos square são do tipo square(linha, coluna, id, estado). Os atributos “linha” e “coluna” representam, como nos octógonos, a posição no tabuleiro e o atributo “estado” representa o estado atual daquele octógono (‘a’ para o jogador A, ‘b’ para o jogador B e ‘o’ para um espaço vazio). O atributo “id” serve de identificador de cada square para mais fácil acesso na lógica do jogo.

```
square(0 ,0,0,x).
square(1 ,0,1,x).
square(2 ,0,2,x).
square(3 ,0,3,x).
square(4 ,0,4,x).
square(5 ,0,5,x).
square(6 ,0,6,x).
square(7 ,1,0,x).
```

Figura 3 - Exemplo da representação dos quadrados

Visualização do Jogo

A visualização do jogo é feita, em modo de texto, da seguinte forma:

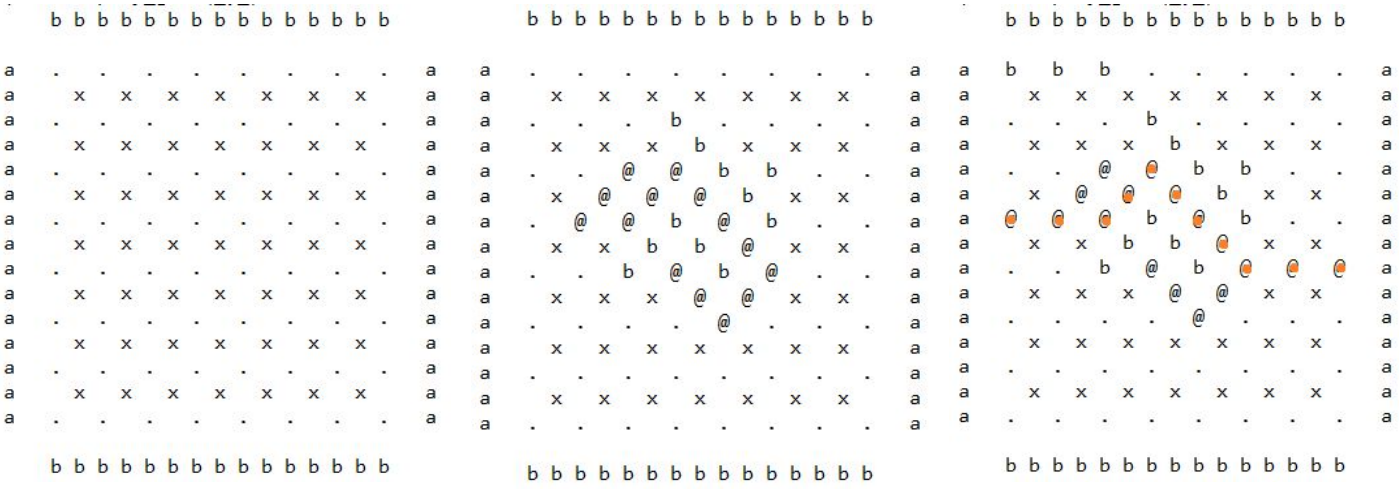

Figura 4 - Exemplo de estado inicial, durante o jogo e estado final (por ordem)

O predicado display_game(+Board, +Player) não necessita de nenhum dos seus parâmetros pois percorre todos os “objetos” octo e square através dos seus índices de linha e coluna. Este predicado começa por escrever a primeira linha de b’s e chama o predicado oformatBoard(+Linha) que imprime, recursivamente, todas as linhas com o auxílio dos predicado oformatLine(+Linha, +Coluna) e oformatSquare(+Linha, +Coluna) que por sua vez imprimem recursivamente os elementos de cada linha. Por fim imprime a última linha de b’s.

```
oformatLine(X,7):-
    octo(X,7,C,_,_,_,_), format('~a   ', [C]).

oformatLine(X,Y):-
    octo(X,Y,C,_,_,_,_), format('~a   ', [C]),
    Y1 is Y+1, Y1 < 8, oformatLine(X,Y1).

oformatBoard(7):-
    write('@   '),oformatLine(7,0), write('@'), nl.
```

```
oformatBoard(X):-
    write('@   '), oformatLine(X,0), write('@'), nl,
    X1 is X+1, X1 < 8,
    write('@   '),oformatSquare(X,0),write('    @'), nl,
    oformatBoard(X1).

oformatSquare(S,6):-
    square(_,S,6,C), format('  ~a ', [C]).

oformatSquare(S,R):-
    square(_,S,R,C), format('  ~a ', [C]),
     R1 is R+1, R1 < 7, oformatSquare(S,R1).

display_game(+Board,+Player):-
    write('    b b b b b b b b b b b b b b b'),nl,nl,
    oformatBoard(0), nl,
    write('    b b b b b b b b b b b b b b b').
```

Nota: deve ser consultado o ficheiro ‘squex.pl’ para aceder a toda a informação.

Bibliografia

BoardGameGeek. ‘Squex’. Accessed 18 October 2019. https://boardgamegeek.com/boardgame/279483/squex.