Programação Funcional e em Lógica

2022 / 2023

Identificação do Trabalho e Grupo

T02_Wali4:

- Diogo André Pereira Babo - up202004950 (50%)
- João Luis Pimenta da Silva Oliveira - up202004407 (50%)

Instalação e Execução

Para a execução do jogo inicialmente é necessária a consulta do ficheiro board.pl, depois apenas é preciso a chamada ao predicado **play/0**.

Descrição do Jogo

Wali trata-se de um jogo de tabuleiro, com dimensões de 5x6, em que cada jogador tem 12 respectivas peças (brancas ou pretas). O jogo está dividido em duas fases:

Drop Phase - Uma vez que o tabuleiro inicialmentese encontra vazio, nesta fase cada jogador tem o objetivo de colocar o maior número de peças no tabuleiro. No entanto, é preciso ter em conta que ao colocar a peça, esta não pode ser adjacente a nenhuma peça do mesmo jogador. Quando um jogador não consegue colocar mais nenhuma peça é obrigado a passar a sua vez. Esta fase acaba quando as peças forem todas colocadas, ou, quando nenhum dos jogadores conseguir colocar mais nenhuma peça.

Move Phase - Nesta fase a cada turno o jogador conseguemover uma peça para um lugar adjacente livre. Caso esse mesmo movimento gere um three-in-a-row, apenas 3, (numa coluna ou numa linha), então ele pode remover uma peça do adversário à escolha. O jogo acaba quando um dos jogadores fica sem peças.

Lógica do Jogo

Representação interna do estado do jogo

Para armazenar a informação do jogo foi utilizada uma lista de listas com inteiros (0, 1 & 2) para representar as peças no tabuleiro ou ausência das mesmas. Sendo que 0 representa um espaço vazio e 1/2 as peças dos jogadores.

```
 1. Estado Inicial (Matriz preenchida c/ 0 )

Board = [
        [0, 0, 0, 0, 0, 0],
        [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. Estado Final (Matriz preenchida c/ peças de 1 jogador)

Board = [
        [1, 1, 1, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0]
      ].
```

Visualização do estado de jogo

A nível de visualização do jogo, é o predicado **display_game(+Board)** que trata da representação do board. Board este que tem o tamanho proposto pelo jogo, 5x6. Este predicado essencialmente percorre as linhas do board e por cada iteração chama o predicado **display_row(+Row)** que dependendo da célula da matriz dá print para o ecrã da peça pretendida.

(Sendo que as peças do player 1 são representadas por um **X**, as peças do player 2 por um **O** e os espaços vazios por um **' '**).

Quanto aos menus, o input é validado e no caso de o utilizador dar o input de uma forma errada este volta a ser pedido, apresentando uma mensagem de erro. O input no jogo para escolher ou mover uma peça é da seguinte forma: **'A2'.**

Execução de Jogadas

Na fase inicial do jogo, as jogadas são realizadas através do seguinte predicado: **putPiece(+Player, +Board, +Row, +Col, -TempBoard)**, contudo antes deste predicado ser chamado é verificada se a jogada é válida através do predicado **checkDrop(+Player, +Board, +Row, +Col)**, que verifica que para aquela coordenada se não há nenhuma peça adjacente do mesmo jogador. No final da jogada, o novo estado do jogo é retornado em **TempBoard**.

Na move phase, as jogadas são realizadas através do predicado proposto no enunciado **move(+Player, +Board, -NewBoard, -Row, -Col)**, em que é passado o novo estado do jogo emNewBoard. Neste predicadoé pedido ao utilizador a peça para mover e a direção, a função verifica se a jogada é válida (ver se o destino é um lugar vazio e se a peça pedida para mover é efetivamente do jogador). Se a jogada for válida, é chamado o predicado **checkThreeMove(+NewBoard, +Player, +Row, +Col)** para verificarse a jogada produziu algum 3-in-a-row, caso isto se verifique então é pedido ao jogador para escolher uma peça do adversário para ser removida.

Lista de Jogadas Válidas

Neste jogo para um jogada ser válida tem que cumprir seguintes requisitos:

Na **Drop Phase**, na posição escolhida não pode haver nenhuma peça do mesmo jogador adjacente (vertical ou horizontal). As jogadas válidas são dadas pelo predicado **findDropAll(+Player, +Board, -Res)**, em queResé uma lista com as jogadas possíveis.

Na **Move Phase**, a peça escolhida tem que se conseguir mover para um espaço vazio adjacente. As jogadas válidas são dadas pelo predicado **valid_moves(+Board, +Player, -ListOfMoves)** em que ListOfMoves vai ser uma lista de listas, sendo quecada sublista tem a posição da peça que pode ser movida e as as posições para onde se pode mover.

Final de Jogo

O final do jogo é verificado através do predicado **game_over**. O jogo acaba quando um dos jogadores fica sem peças. No final é mostrada uma mensagem no terminal contendo o jogador que venceu.

Jogada do Computador

O computador tem 2 níveis de dificuldade, que com base na situação que se encontram se comportam de maneira diferente.

O **primeiro** nível é o nível fácil, que consiste em escolher uma jogada aleatória de todas as jogadas válidas.

O **segundo** nível é o nível difícil que consiste na implementação de uma estratégia ambiciosa. Esta estratégia na 'Drop Phase' começa por colocar as pedras o mais junto possível umas das outras. Na 'Move Phase' o algoritmo inicia por analisar se alguma peça ao ser movida pode fazer um 3 em linha. No caso de se verificar, é feito esse movimento e a dá se prioridade a

remover peças adversárias que no imediato fazem também 3 em linha, no inverso é selecionada a peça mais longe da origem e move-se em direção à origem, com o objetivo de aproximar as peças todas para o mesmo ponto de forma a aumentar a probabilidade de se verificar um 3 em linha.

É possível jogar contra o computador em qualquer um dos modos e também é possível o modo Computador vs Computador (Hard vs Hard, Easy vs Easy & Easy vs Hard).

Conclusão

O projeto teve como objetivo aplicar o conhecimento adquirido nas aulas teóricas e práticas, da 2 ª parte da unidade curricular de Programação Funcional e em Lógica.

Ao longo da realização do projeto fomos desenvolvendo o pensamento diferente da maioria das linguagens com que já trabalhamos até agora. Pensamos que a nossa implementação cumpre os requisitos, já que conseguimos desenvolver o jogo com sucesso, apesar de haver espaço para melhorias no código, como por exemplo na segunda dificuldade do BOT.

Não foi possível arranjar uma função concreta para avaliar o estado do board, uma vez que, a definição de uma jogada melhor comparativamente a outra, com base em diferentes parâmetros para além de realizar um 3 em linha ou um 2 em linha, não é trivial. Isto limitou que, a estratégia ambiciosa utilizada fosse linear e previsível e incapaz de ter em conta o futuro.

Bibliografia

Não foram encontradas muitas informações sobre o jogo excepto o link que estava no enunciado do trabalho prático.
https://www.di.fc.ul.pt/~jpn/gv/wali.htm