Identificação do Trabalho e do Grupo

O nosso trabalho é baseado no jogo “Jeson Mor” (Jeson Mor_4), realizado por:
• Diogo Luís Araújo de Faria – up201907014 – 50%
• Tiago André Batista Rodrigues – up201906807 – 50%

Instalação e execução

Para a utilização do jogo, apenas é necessário fazer “consult” no SICStus Prolog no ficheiro “Game.pl”, sendo apenas necessário para a execução do jogo os seguintes passos:

1. Executar o predicado “play.”;
2. Escolher o tipo de jogo, desde “a” até “i”;
3. Escolher o tamanho do tabuleiro, maior que 5 e de tamanho de ímpar;
4. Nas jogadas feitas por humanos, selecionam-se as posições de inicio de fim para uma jogada.

Descrição do Jogo

O Jeson Mor é um jogo derivado do xadrez, sendo que é jogado por 2 jogadores, um com peças pretas e outro com peças brancas e cada jogador começa com uma fila inteira de cavalos, os quais apenas podem realizar movimentos em ‘L’, tal como em xadrez. De forma a ganhar o jogo, é necessário que uma das peças de um jogador ocupe a posição central do tabuleiro e, em jogadas seguintes, a retire da mesma. Por outro lado, também é possível ganhar, efetuando uma captura sobre todas as peças do adversário. Usamos como referência https://en.wikipedia.org/wiki/Jeson_Mor e https://boardgamegeek.com/boardgame/37917/jeson-mor .

Lógica de Jogo

Representação interna do estado do jogo

O estado do jogo é representado utlizando:

```
game_state(Size, TurnNo, Board, Player, Player1Type, Player2Type)
```

• Size - representa o tamanho do lado do tabuleiro, sendo que vai ser sempre quadrado (maior que 5 e ímpar);
• TurnNo - representa o turno currente;
• Board - representa o board como uma lista de listas, sendo cada peça representada por um 'W' ou 'B', utilizando letra minúscula 'w' ou 'b' quando se move uma peça do centro do tabuleiro, ou seja, numa vitória.
• Player - representa o jogador corrente (1 ou 2);
• Player1Type - representa o tipo do jogador 1 (human, computer-0 ou computer-1);
• Player2Type - representa o tipo do jogador 2 (human, computer-0 ou computer-1).

No inicio do jogo, uma possibilidade de game_state apresenta os seguintes valores, tendo-se escolhido o modo 'e' (human/computer-1):
• Size - 9;
• TurnNo - 0;
• Board - [['W','W','W','W','W','W','W','W','W'], [' ',' ',' ',' ',' ',' ',' ',' ',' '], [' ',' ',' ',' ',' ',' ',' ',' ',' ',' '], [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '], [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '], [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '], [' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ',' '], ['B','B','B','B','B','B','B','B','B']];
• Player - 1;
• Player1Type - human;
• Player2Type - computer-1.

Durante o jogo, uma possibilidade de game_state apresenta os seguintes valores:

• Size - 9;
• TurnNo - 34;
• Board - [[,W, , , , , ,],[, ,W, , , , ,W,W],[, , ,W,W, , ,W,],[, , , ,W, , ,],[, ,B, , , , ,],[, , , ,B, , ,B,],[, , , , , ,B,B,],[W, , , , , , ,],[B,B, ,B, , , , ,B]];
• Player - 2;
• Player1Type - computer-0;
• Player2Type - computer-0.

No final do jogo, uma possibilidade de game_state apresenta os seguintes valores, sendo que o jogador 1 ganhou:

• Size - 9;
• TurnNo - 5;
• Board - [[W,W,W,W, ,W,W,W,W],[, , , , , , , ,],[, , ,w, , , ,],[, , , , , , , ,],[, , , , , , , ,],[, , , , , , , ,],[, , , , , , ,B,],[, , , , , , , , ,],[B,B,B,B,B,B,B, ,B]];
• Player - 1;
• Player1Type - computer-1;
• Player2Type - computer-0.

Visualização do estado de jogo

O predicado de visualização do jogo é 'display_game(+GameState)', que é executado da seguinte forma:

1. Display de fila de letras utilizada para coordenadas, utilizando um predicado auxiliar 'letter_display(+Acc, +Size)', utlizando o código ASCII de cada letra a começar em '97' (letra 'a') e incrementando '1' utlizando o 'Acc' até que o acumulador seja igual a 'Size', utlizando vários 'write' tanto para as letras em si, como para os caracteres entre elas;
2. Display do resto do tabuleiro é feito utilizando o predicado 'display_game_aux(+GameState, +Line)', em que 'Line' é a linha no momento, sendo utilizada para imprimir uma coluna de números no lado direito do tabuleiro, utilizados em conjunção com as letras para coordenadas. As peças são escritas no ecrã percorrendo o tabuleiro lista por lista, utilizando 'row_display(+Row, +Line)', para cada lista em especifico. Estas utilizam também o predicado 'write';
3. Entre cada linha escrita no ecrã, utliza-se um predicado auxiliar 'display_between(+Acc, +Size)', inicializado com 'Acc' com 0, que imprime delimitações.

O predicado de menus criados é 'read_game_type(-Player1Type, -Player2Type)':

1. Este predicado recebe input do ulizador relativo ao tipo de jogo. Para isto, é escrito no ecrâ, utilizando 'write', as opções possíveis, sendo entre 'a' e 'i', recebidos utilizando 'read'. De forma a verificar se o input é válido, verifica-se o código ASCII do caracter escrito.

O predicado de tamanho do tabuleiro é 'read_board_size(-Size)':

1. Este predicado recebe tamanho do tabuleiro a ser criado. Este é recebido com 'read' e verificado vendo se é maior ou igual a 5 e ímpar.

O predicado 'read_move_input(+GameState, -Move)' recebe um movimento do jogador da seguinte forma:

1. É impresso no ecrã o turno currente e o jogador;
2. De seguida, é impresso um pedido para a posição da peça que o jogador pretende mover, utilizando 'write' e 'read' e depois a posição final dessa mesma peça;
3. No fim, verifica-se se este input é valido e se o movimento é correto, utilizando 'valid_positions(+Board, +Player, +Move)'.

Execução de Jogadas

O predicado de execução de jogadas 'move(+GameState, +Move, -NewGameState)' é executado da seguinte forma:

1. Valida o movimento, chamando a função 'valid_positions(+Board, +Player, +Move)', que valida cada posição do movimento e se esse mesmo pode ser realizado;
2. De seguida, utiliza-se 'execute_move(+Board, +Player, +Move, -NewBoard, +Center)', sendo que 'Center' é a posição central do tabuleiro, recebido por 'center_board(+Size, -Center)'. Para cada jogador, é verificado se a posição inicial é a do centro do tabuleiro, de forma a efetuar o movimento, em que se substitui a posição inicial por ' ' e a final pela letra minúscula correspondente. Se não for a posição central, simplesmente substitui a incial por ' ' e a final pela letra maiúscula correspondente;
3. Por fim, aumenta-se 1 ao turno.

Final do Jogo

O predicado de final do jogo 'game_over(+GameState, -Winner)' é executada da seguinte forma:

1. Procura uma peça 'w' ou 'b', que identifica o final de jogo e ganha o jogador que tenha a letra minúscula.
2. A seguir vê a quantidade de peças de cada jogador, caso tenha 0 peças, o outro jogador ganha.
3. Caso contrário, falha.

Lista de Jogadas Válidas

O predicato de lista de jogadas válidas 'valid_moves(+GameState, -ListOfMoves)' é composto por um 'findall()' onde se inicializam valores para posições de movimentos, utilizando o predicado 'between' 4 vezes consecutivas, seguido do predicado 'move', sendo que se este não falhar, então o movimento é adicionada à lista de jogadas.

Avaliação do Estado do Jogo

O predicado de avaliação do estado do jogo 'value(+GameState, -Value)' é executada do seguinte forma:

1. É dividido em vários predicados que constituem uma parte do valor que junto dá -1, para ajudar quando usado em 'choose_move'. Apenas no caso de vitória no próxima jogada que o value se torna -1000 para ter prioridade.
2. O predicado 'value_number(+GameState, -V1)' que contabiliza o número de peças que temos em relação á quantidade de peças inicial. A variável V1 varia entre [-0.15, 0].
3. O predicado 'value_piece_center(+GameState, -V2)' que contabiliza a distância ao centro de cada peça em função da distância máxima para o centro. A variável V2 varia entre [-0.25, 0].
4. O predicado 'value_pointing_center(+GameState, -V3)' que contabiliza a quantidade de peça a apontar para o centro. A variável V3 varia entre [-0.20, 0]
5. O predicado 'value_piece_difference(+GameState, -V4)' que contabiliza a diferença de peças entre o jogador e o adversário em função da quantidade das peças. A variável V4 varia entre [-0.20, 0].
6. O predicado 'value_center(+GameState, -V5)' que contabiliza -0.20 caso tenha uma peça no centro e 0 caso o jogador não tenha. A variável V5 apenas pode ter valores -0.20 ou 0.

Jogada de Computador

O predicado para a jogada de computador é 'choose_move(+GameState, +PlayerType, -Move)', onde PlayerType indica o tipo de jogador, sendo que se for o computador existe computer-0 e computer-1, para bot random ou greedy, respetivamente:

1. Independentemente do nivel do bot, o predicado começa por chamar 'valid_moves(+GameState, Moves)', que obtêm uma lista com todos os movimentos válidos possíveis;
2. De seguida, escolhe-se o movimento, dependendo do nivel do bot. Se for nivel 0, ou seja, bot random, utliza-se 'random_member' para escolher um movimento aleatório da lista de movimentos. No entanto, se for nivel 1, ou seja, bot greedy, utliza-se um 'setof', que vai iterar sobre a lista de movimentos possíveis e chamando a função 'move' para os fazer, sendo que no fim avalia o estado do jogo no 'GameState' resultante. Como o predicado 'value' apresenta o valor do jogo em número negativo (sendo que quanto menor o número do valor, maior é o valor real do jogo) e o 'setof' ordena a lista de forma ascendente, o primeiro elemento da lista vai ser o movimento escolhido;
3. Finalmente, apenas se apresenta no ecrã o movimento que o computador vai efetuar, utilizando o predicado 'write_computer'.

Conclusões

Concluimos o trabalho com sucesso, sendo que uma possível melhoria era o cálculo de valores e um bot mais inteligente a utilizar maior profundidade.

Bibliografia

Para descobrir e conhecer as regras do jogo usámos:
https://en.wikipedia.org/wiki/Jeson_Mor
https://boardgamegeek.com/boardgame/37917/jeson-mor