Moxie Prolog

Identificação do trabalho e do grupo

O jogo escolhido pelo nosso grupo foi o Moxie

Trabalho foi realizado por:

- Gonçalo Carvalho Marques -up202006874
- David Oliveira Espinha Marques -up201905574

Instalação e Execução

Para instalar o jogo baste consultar o ficheiro moxie.pl, todas as dependências necesssárias ao programa serão carregadas.
Para executar o jogo tem de se executar o predicado start/0.

Descrição do jogo

As regras do jogo são simples:
O jogo é jogado num tabuleiro de 4 por 4 e cada jogador tem inicialmente 8 peças. Na sua jogado o jogador pode fazer uma de três coisas, mover uma das suas peças para uma posição adjacente que esteja vazia, por uma nova peça no tabuleiro ou capturar uma peça do adversário fazendo a sua peça saltar por cima da do adversário e cair numa casa vazia imediatamente a a seguir a peça capturada. Caso o jogador possa executar uma catpura esta é obrigatória e é possível executar multiplas capturas numa só jogado.
O objetivo do jogo é conseguir fazer três em linha em qualquer direção ou capturar 6 peças do adversário.
Mais informações sobre o jogo podem ser encontradas aqui:
https://www.di.fc.ul.pt/~jpn/gv/moxie.htm

Lógica do jogo

Incio do Jogo

Para dar início ao jogo em si o predicado play/0 é chamado este chama os predicados gamemode/1 size/1 que podem ser modifcados no menu, e inicializa um tabuleiro com as caracteristicas definidas. Se o jogador não definir gamemode nem board size os valores default de h/h e 4 são utilizados.

Representação interna do estado do jogo

Durante a execução do jogo são guardades internamente vários dados importatnes ao jogo, no predicado game_state(-turnNum,-P1State,-P2State,-Board) são guardados o número do turno, a pontuação de ambos os jogadores e o estado do tabuleiro.
O tabuleiro é representado por uma lista de listas de dimensão size x size, cada elemento desta matriz representa uma casa do tabuleiro sendo que se o elemento for 0 a casa encontra-se vazia, 1 tem uma peça do jogador 1 e 2 tem uma peça do jogador 2.

Visualização do estado de jogo

O visualização do jogo está ao encargo do predicado display_game(+Gamestate) que recebe o estado do tabuleiro e imprime na consola um versão modificada desta para a leitura do mesmo ser mais facil. Assim caso o predicado enontre um 0 imprime um espaço vazio, se encontrar um 1 imprime uma cruz e se encontrar um 2 imprime um circulo representando assim as peças de ambos os jogadores.
Quando os jogadores iniciam o jogo encontrão o menu do mesmo, este tem várias opções disponiveis:

- Play: Começar a jogar
- Change Gamemode: permite escoclher o modo de jogo entre pvp e pve para escolher o modo de jogo o jogador tem que escrever que tipo de jogador é que quer h para humano, pc1 para o bot no modo facil e pc2 para o no modo díficil bot ex: h/h seria o modo de jogo de humano vs humano
- Change board size: O jogador pode escolher o tamanho do tabuleiro
- Leave: Sair do jogo

Execução de Jogadas

Quando começa a jogado de algum dos utilizadores o predicado player_turn(+Player, +PlayerType) é chamado este predicado verifica se se trata do turno do primeiro o segundo jogador e o tipo de do jogador (se é um bot ou um humano), e dependo destes dois valores chama os predicados adequados.

- Humano: Primeiro faz-se a verificação de que jogados é que o jogador pode executar e mostra-se esta lista ao mesmo, em seguida é lido o input do jogador e caso este seja valido a jogada é executada caso contrario pede-se para o jogador repetir o input
- Bot: Primeiro computa-se todas as jogadas possiveis que o bot pode executar e caso se trate do bot facíl uma jogada aleatória é escolhida recorrendo ao módulo random, caso se trate do bot dificil as jogadas são avaliadas e a melhor é escolhida baseado na pontuação das mesmas.

A escolha das jogadas por parte dos jogadores humanos é feito com o predicado choose_play(+Play,+Player) que recebe o input dos jogadores e chama o predicado respetivo place_piece(+Player,+Row,+Col,+Board),move_piece(+Player,+Row1,+Col1,+Row2,+Col2, +Board) ou eat_piece(+Player,+Row,+Col,+Row2,+Col2, -Row3, -Col3, +Bpard), estes predicados são responsaveis por verificarem se a jogada é válida e modificarem o estado do tabuleiro.

Lista das jogadas válidas

As jogadas válidas podem ser obtidas usando o predicado valid_plays(+Player, +Board, -Plays) que recebendo um jogador e o estado do tabuleiro devolve num vetor todas as jogadas que o jogador pode fazer.

Final do Jogo

Para detetar quando o jogo acaba são usados os predicados game_over(+Winner) e line_win(+Player) que verificam respetivamente se um jogador obteve a pontuação necessária para ganhar ou se fez um três em linha.

Avaliação do tabuleiro

Para conseguirmos avaliar o estado do tabuleiro precisamos de ter em atenção várias métricas tais como o número de peças de cada jogador no tabuleiro, a posição relativa destas peças. O valor de cada jogada pode ser calculado pelo predicado play_value(+Player, +Play, +Board, -Score), que atribui uma pontuação a cada jogada especifica sendo que quanto maior essa pontuação melhor a jogada.

Jogada do Computador

Na jogada do computador primeiro é verificado se o bot está no nivel facil ou dificil (pc1 ou pc2), para os dois casos primeiro obtem-se todas as jogadas validas usando o predicado valid_plays(+Player, +Board, -Plays). Em seguida casa se trate do bot fácil +e escolhida uma jogada aleatória, usando o predicado random_select(?Elem, ?List, ?Rest) da library(random). Caso se trate do bot dificil escolhe-se a melhor jogada possivel usando o predicado best_play(+Player, +Plays, +Board, -BestPlay, -BestValue) que recebe todas as jogadas válidas para um jogador e usando o predicado play_value(+Player, +Play, +Board, -Score) compara as suas pontuações escolhendo a melhor, no caso de várias jogadas terem a pontuação máxima é escolhido uma dessas aleatoriamente.

Conclusões

Ao realizar este projeto tivemos a oportunidade de utilizar uma linguagem que não nos era familiar e descobrir uma paradigma de progamação completamente diferente daqueles a que estamos habituados o que nos fez pensar de formas diferentes.
As maiores limitações do nosso trabalho prendem-se com os predicados de manipulação de listas pois estes não são triviais de fazer em Prolog, assim é muito provavel que os predicados criados por nós para este efeito não se encontrem no melhor estado de otimização.
Algumas melhorias que se poderiam fazer neste projeto seriam a otimização dos predicados falados acima e a adição de mais dificuldades aos bots.

Bibliografia

- https://www.di.fc.ul.pt/~jpn/gv/moxie.htm
- https://www.swi-prolog.org
- Documentação da UC