REPLICA

Trabalho prático 2

Programação Funcional e em Lógica

Turma 14 – Replica_4

Tiago de Pinho Bastos de Oliveira Pinheiro (up202207890) – 50% (desenvolvimento da lógica dos transforms e bots e contribuição nas condições de fim de jogo)

Tiago Grilo Ribeiro Rocha (up202206232) – 50% (desenvolvimento da lógica inicial do jogo e dos movimentos e contribuição nas condições de fim de jogo)

Contribuições: No geral a distribuição do trabalho foi muito idêntica para ambos.

Instalação e execução do jogo ----------

Os ficheiros para o nosso jogo encontram-se presentes no ficheiro zip PFL_TP2_T14_Replica_4.zip. Depois de o transferir, basta descomprimir o ficheiro. Agora, dentro da UI do Sicstus Prolog, escolhemos o diretório src, compilamos o ficheiro game e finalmente executamos com o predicado play/0. Os comandos necessários, após escolher o diretório, são “[game].” (compilar) e “play.” (executar).

Descrição do jogo ----------

Replica é um jogo para dois jogadores jogado num tabuleiro de xadrez com 12 peças brancas e 12 peças pretas. Para preparar o jogo, os jogadores colocam as peças nos cantos opostos do tabuleiro (num quadrado de 2x2 no canto, junto de um quadrado de 2x2 de cada lado). As peças posicionadas diretamente no canto são as peças “rei”. Os jogadores fazem um movimento por turno, sendo o jogo iniciado pelas peças brancas.

Em cada turno, os jogadores podem avançar, saltar ou transformar. Todos os movimentos (até mesmo as capturas) devem ser realizados "para a frente" (numa das três direções em direção canto oposto). Nos movimentos de avanço e salto, se já houver uma peça adversária na casa de destino, esta é capturada e substituída.

- **Avançar**: a peça move-se uma casa para a frente.

- **Saltar**: a peça move-se numa linha reta para a frente, saltando por cima de peças aliadas até alcançar uma casa desocupada ou com uma peça inimiga.
- **Transformar**: uma peça aliada que não seja um rei, mas que esteja na linha de visão de um rei aliado, é transformada numa peça rei. Apenas peças inimigas bloqueiam a linha de visão.

O jogo termina quando um jogador consegue levar um rei aliado qualquer até ao canto oposto do inimigo ou ao capturar um rei inimigo qualquer**.**

Considerações para extensões do jogo

O nosso jogo adiciona as seguintes regras extra ao jogo base:

- Os reis podem andar para todas as direções.
- Os reis aliados podem transformar peças em quaisquer direções em que se possa movimentar desde que a linha de visão não esteja bloqueada

O jogador pode ainda escolher utilizar ou não a seguinte regra:

- Cada jogador pode apenas transformar peças aliadas 2 vezes.

Lógica do jogo ----------

Representação da configuração do jogo

A configuração do jogo inclui toda a informação necessária para inicializar o jogo, sendo utilizada pelo predicado initial_state/3, que usa, mais precisamente, o tipo de jogadores e o layout do tabuleiro inicial.

Representação do estado interno do jogo

O estado interno do jogo retrata a configuração do tabuleiro, do jogador atual e os tipos de jogadores no jogo a decorrer.

O tabuleiro é representado como uma lista de listas que apresenta um tabuleiro 8x8, com coordenadas (X, Y). Cada casa pode ter as seguintes informações:

- empty: espaço sem quaisquer peças (representado por '**.**');
- white: peça normal do jogador branco (representado por '**W**');
- black: peça normal do jogador preto (representado por '**B**');
- kingw: rei branco (representado por '**o**');
- kingb: rei preto (representado por '*****');

O jogador atual é representado ou por “white” ou por “black”. Já os tipos de jogador, podem ser “humanos”, “bot_random” ou “bot_greedy”.

Exemplos de estados de jogo

Estado inicial

```
Choose an option (1-5): |: 1.

   +-----------------------+
8  |  .  .  .  .  B  B  B  *  |
7  |  .  .  .  .  B  B  B  B  |
6  |  .  .  .  .  .  .  B  B  |
5  |  .  .  .  .  .  .  B  B  |
4  |  W  W  .  .  .  .  .  .  |
3  |  W  W  .  .  .  .  .  .  |
2  |  W  W  W  W  .  .  .  .  |
1  |  o  W  W  W  .  .  .  .  |
   +-----------------------+
      1  2  3  4  5  6  7  8

white player's turn.
```

Estado intermediário

```
Bot chooses action: [transform,2,7,2,2]

   +-----------------------+
8  |  .  .  .  .  .  .  .  .  |
7  |  .  o  B  B  .  .  .  B  |
6  |  .  .  .  .  W  B  .  .  |
5  |  .  .  .  B  .  .  .  .  |
4  |  .  .  .  B  B  .  B  .  |
3  |  o  W  *  o  .  .  B  .  |
2  |  W  o  .  W  B  .  .  .  |
1  |  .  .  W  .  W  .  .  .  |
   +-----------------------+
      1  2  3  4  5  6  7  8

black player's turn.
```

Estado final

```
Bot chooses action: [5,5,6,5]

   +-----------------------+
8  |  .  *  .  .  .  *  .  .  |
7  |  .  .  .  B  .  .  W  .  |
6  |  .  *  .  W  B  .  *  .  |
5  |  .  .  .  .  .  W  .  .  |
4  |  B  W  .  .  .  *  .  W  |
3  |  .  o  .  .  *  .  .  .  |
2  |  .  W  .  .  .  W  W  W  |
1  |  .  .  .  .  .  .  .  .  |
   +-----------------------+
      1  2  3  4  5  6  7  8

Game Over! Winner: white
```

Representação dos movimentos

Na nossa implementação do jogo os movimentos têm em consideração o estado do tabuleiro, o jogador atual, os tipos dos dois jogadores (humano ou bot), o novo estado após a jogada e uma variável que verifica se o jogo termina com a jogada.

Internamente um movimento regular tem em conta 4 números. Os dois primeiros representam a posição inicial e os dois últimos a posição do deslocamento. A validade deste movimento é verificada com a função valid_move/3.

Por outro lado, os “transform moves” têm a palavra-chave no início, seguida por 4 números. Os dois primeiros representam as coordenadas do rei que vai ser

utilizado para transformar a peça com as coordenadas dos dois últimos. Estas jogadas são validadas pela função valid_transform/3.

Após a validação dos movimentos, estes são realizados com o predicado make_move/3 que cria um novo estado do jogo após o movimento.

Interação do usuário

Ao iniciar o jogo, o usuário depara-se com o nosso menu, onde pode escolher quaisquer um dos seguintes modos de jogo:

```
========================================
           Welcome to Replica
========================================
1. Play Human vs Human
2. Play Human vs Computer
3. Play Computer vs Human
4. Play Computer vs Computer
5. Rules & How to play
6. Exit
10. [DEMO] Midgame scenario
20. [DEMO] Endgame scenario
========================================
Choose an option (1-6):
```

Dentro do jogo, para além dos movimentos, o usuário tem várias opções de comandos que são visíveis usando “[commands].”, como por exemplo:

- Verificar jogadas possíveis: [validmoves].
- Verificar o ‘score’ de cada jogada possível: [scoredmoves].
- Verificar o ‘score’ geral de cada jogador: [value].
- Desistir e dar vitória ao outro jogador: [forfeit].

Conclusões

Ao finalizar o desenvolvimento, em Prolog, do jogo “Replica” concluímos que apesar de alguns problemas, achamos que conseguimos atingir expectativas e até adicionar alguns toques pessoais ao jogo. Tivemos o cuidado de tratar graciosamente com todos os possíveis erros com inputs do usuário, de forma a não terminar o programa abruptamente. Uma dificuldade notável foi a interpretação inicial do jogo, uma vez que este não é muito conhecido ou falado. É também de notar que o bot_greedy foi outra das nossas complicações, infelizmente maioritariamente devido à falta de tempo pessoal para trabalhar nele. Com isto tudo em mente, sentimos que aprendemos muito e reforçamos os conhecimentos das aulas práticas e teóricas.

Bibliografia

https://boardgamegeek.com/boardgame/427267/replica