Trabalho Prático 2

Aplicação em Prolog para um Jogo de Tabuleiro

Descrição

Objetivo: Implementar um jogo de tabuleiro, para dois jogadores, em linguagem Prolog. Um jogo de tabuleiro caracteriza-se pelo tipo de tabuleiro e de peças, pelas regras de movimentação das peças (jogadas possíveis) e pelas condições de terminação do jogo com derrota, vitória ou empate. O jogo deve permitir três modos de utilização: Humano/Humano, Humano/Computador e Computador/Computador. Devem ser incluídos pelo menos dois níveis de jogo para o computador. Deve ser construída uma interface adequada com o utilizador, em modo de texto.

Condições de Realização

Constituição dos Grupos: Grupos de 2 estudantes, inscritos na mesma turma teórico-prática. Excepcionalmente, e apenas em caso de necessidade, podem aceitar-se grupos de 3 elementos.

Escolha de Grupos e Temas: Os grupos deverão ser indicados na atividade a disponibilizar para o efeito no Moodle a partir das 16:00 do dia 18 de novembro de 2022. Numa primeira fase, **u**m dos elementos do grupo deverá fazer a escolha do grupo; numa segunda fase, o segundo elemento juntar-se-á ao grupo. Cada jogo pode ser escolhido por um máximo de **6** grupos, de modo a garantir que todos os jogos são igualmente selecionados. No final deste enunciado encontra-se uma lista com os jogos selecionáveis.

Prazos: A entrega do trabalho (código-fonte e ficheiro readme) deverá ser realizada até ao final do dia 2 de janeiro de 2023, na atividade a disponibilizar para o efeito no Moodle, e com demonstrações realizadas na semana de 2 a 6 de janeiro de 2023. As demonstrações serão combinadas com o docente de cada turma prática.

Pesos das Avaliações: Ver ficha da Unidade Curricular no SIGARRA.

Linguagens e Ferramentas: O jogo deve ser desenvolvido em SICStus Prolog versão 4.7.1, e deve ser garantido o seu funcionamento em Windows e Linux. Caso seja necessária alguma configuração (para além da instalação padrão do *software*), ou seja usada uma fonte diferente da fonte por omissão, isso deverá estar expresso no ficheiro README que deverá ainda incluir os passos necessários para configurar e/ou instalar as componentes necessárias (em Windows e Linux). A impossibilidade de testar o código desenvolvido resultará em penalizações na avaliação. Deve ter o cuidado de nomear os predicados usados conforme pedido na descrição abaixo. Todo o código deve ser devidamente comentado.

Avaliação

Cada grupo deve entregar um relatório em formato README e o código-fonte desenvolvido, bem como realizar uma demonstração da aplicação. A submissão deverá ser em formato ZIP na plataforma *Moodle*, e o nome do ficheiro deverá ser:

PFL_TP2_**TX_#GRUPO**.ZIP

em que **TX** indica a turma prática (ex. T06 para a turma 3LEIC06), e **#GRUPO** é a designação do grupo. Exemplo: PFL_TP2_T06_Xadrez4.ZIP

O ficheiro ZIP deverá conter um ficheiro **README** e o **código-fonte** PROLOG, o qual deverá ser **devidamente comentado**.

O ficheiro README deve ter a seguinte estrutura:

- **Identificação do trabalho** (jogo) **e do grupo** (designação do grupo, número e nome completo de cada um dos elementos), assim como indicação da contribuição (em percentagem, somando 100%) de cada elemento do grupo para o trabalho;
- **Instalação e Execução**: incluir todos os passos necessários para correta execução do jogo em ambientes Linux e Windows (para além da instalação do SICStus Prolog 4.7.1).
- **Descrição do jogo**: descrição sumária do jogo e suas regras (até 350 palavras); devem incluir ainda ligações usadas na recolha de informação (página oficial do jogo, livro de regras, etc.);
- **Lógica do Jogo:** Descrever (não basta copiar o código fonte) o projeto e implementação da lógica do jogo em Prolog. **O predicado de início de jogo deve ser *play/0***. Esta secção deve ter informação sobre os seguintes tópicos (até 2400 palavras no total):
    - **Representação interna do estado do jogo:** indicação de como representam o estado do jogo, incluindo tabuleiro (tipicamente usando lista de listas com diferentes átomos para as peças), jogador atual, e eventualmente peças capturadas e/ou ainda por jogar, ou outras informações que possam ser necessárias (dependendo do jogo). Deve incluir exemplos da representação em Prolog de estados de jogo inicial, intermédio e final, e indicação do significado de cada átomo (ie., como representam as diferentes peças).
    - **Visualização do estado de jogo:** descrição da implementação do predicado de visualização do estado de jogo. Pode incluir informação sobre o sistema de menus criado, assim como interação com o utilizador, incluindo formas de validação de entrada. O predicado de visualização deverá chamar-se ***display_game(+GameState)***, recebendo o estado atual do jogo (que inclui o jogador que efetuará a próxima jogada). Serão valorizadas visualizações apelativas e intuitivas. Serão também valorizadas representações de estado de jogo e implementação de predicados de visualização flexíveis, por exemplo, funcionando para qualquer tamanho de tabuleiro, usando um predicado ***initial_state(+Size, -GameState)*** que recebe o tamanho do tabuleiro como argumento e devolve o estado inicial do jogo.
    - **Execução de Jogadas:** Validação e execução de uma jogada, obtendo o novo estado do jogo. O predicado deve chamar-se ***move(+GameState, +Move, -NewGameState)***.
    - **Lista de Jogadas Válidas:** Obtenção de lista com jogadas possíveis. O predicado deve chamar-se ***valid_moves(+GameState, +Player, -ListOfMoves)***.
    - **Final do Jogo:** Verificação do fim do jogo, com identificação do vencedor. O predicado deve chamar-se ***game_over(+GameState, -Winner)***.
    - **Avaliação do Tabuleiro:** Forma(s) de avaliação do estado do jogo. O predicado deve chamar-se ***value(+GameState, +Player, -Value)***.
    - **Jogada do Computador:** Escolha da jogada a efetuar pelo computador, dependendo do nível de dificuldade. O predicado deve chamar-se ***choose_move(+GameState, +Player, +Level, -Move)***. O nível 1 deverá devolver uma jogada válida aleatória. O nível 2 deverá devolver a melhor jogada no momento (algoritmo greedy), tendo em conta a avaliação do estado de jogo.
- **Conclusões:** Conclusões do trabalho, incluindo limitações do trabalho desenvolvido (*known issues*), assim como possíveis melhorias identificadas (*roadmap*) (até 250 palavras);

- **Bibliografia:** Listagem de livros, artigos, páginas Web e outros recursos usados durante o desenvolvimento do trabalho.

O código-fonte desenvolvido deverá estar dentro de um diretório denominado ***src***, e deverá estar **devidamente comentado**. O predicado principal ***play/0*** deve dar acesso ao menu de jogo, que permita configurar o tipo de jogo (H/H, H/PC, PC/H, PC/PC), níveis de dificuldade a usar nos jogadores artificiais, entre outros possíveis parâmetros, e iniciar o ciclo de jogo.

Pode ainda incluir uma ou mais **imagens** ilustrativas da execução do jogo, mostrando um estado de jogo inicial, e possíveis estados intermédios e final (estes estados de jogo podem ser codificados diretamente no ficheiro de código para esta demonstração da visualização do estado de jogo, usando predicados semelhantes ao predicado ***initial_state/2***).

Problemas (Jogos) Propostos

Os jogos a implementar são jogos de tabuleiro para dois jogadores em que não existe a influência do fator sorte no decorrer do jogo. Os jogos não incluem dados nem sorteios de qualquer tipo ou informação inicialmente escondida.

Jogos propostos:

1. 369 - https://www.di.fc.ul.pt/~jpn/gv/369.htm
2. Barca - https://boardgamegeek.com/boardgame/69347/barca
3. Brood - https://boardgamegeek.com/boardgame/368284/brood
4. Center - https://boardgamegeek.com/boardgame/360905/center
5. Freedom - https://www.iggamecenter.com/en/rules/freedom
6. Hadron - https://boardgamegeek.com/boardgame/368118/hadron
7. Moxie - https://www.di.fc.ul.pt/~jpn/gv/moxie.htm
8. Nex - https://www.iggamecenter.com/en/rules/nex
9. Qawale - https://www.hachetteboardgames.com/products/qawale
10. Ski Jumps - https://www.di.fc.ul.pt/~jpn/gv/skijump.htm
11. Stlts - https://www.iggamecenter.com/en/rules/stlts
12. Storm - https://www.di.fc.ul.pt/~jpn/gv/storm.htm
13. Taacoca - https://www.iggamecenter.com/en/rules/taacoca
14. Tako Judo - https://boardgamegeek.com/boardgame/29291/tako-judo
15. Ugly Duck - https://www.di.fc.ul.pt/~jpn/gv/uglyduck.htm
16. Wali - https://www.di.fc.ul.pt/~jpn/gv/wali.htm
17. Wana - https://boardgamegeek.com/boardgame/364012/wana
18. Yote - https://www.di.fc.ul.pt/~jpn/gv/yote.htm
19. Yoxii - https://boardgamegeek.com/boardgame/361084/yoxii