Trabalho Prático 2

Descrição

Objetivo: Implementação de um jogo de tabuleiro para dois jogadores em linguagem Prolog. Um jogo de tabuleiro caracteriza-se pelo tipo de tabuleiro e peças, regras de movimentação das peças (jogadas possíveis) e condições de terminação do jogo com derrota, vitória ou empate. O jogo deve permitir vários modos de utilização (com jogador humano e/ou artificial), apresentando uma interface com o utilizador em modo de texto.

Condições de Realização

Constituição dos Grupos: Grupos de 2 estudantes, inscritos na mesma turma teórico-prática. Excecionalmente, 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 10 de dezembro de 2021. Numa primeira fase, um dos elementos do grupo deverá fazer a escolha do grupo; numa segunda fase, o segundo elemento juntar-se-á ao grupo. Cada tema pode ser escolhido por um máximo de 5 grupos, de modo a garantir que todos os temas são igualmente selecionados. No final deste enunciado encontra-se uma lista com os possíveis temas de trabalho.

Prazos: A entrega do trabalho (código-fonte e ficheiro readme) deverá ser realizada até ao final do dia 23 de janeiro de 2022, na atividade a disponibilizar para o efeito no *Moodle*, e com demonstrações realizadas na semana de 24 a 28 de janeiro de 2022. 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, 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 necessário 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 ficheiro 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. T1 para a turma 3LEIC01), e **#GRUPO** é a designação do grupo. Exemplo: `PFL_TP2_T6_Xadrez4.ZIP`

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

O ficheiro README deve ter a seguinte informação:

- **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);
- **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 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 de jogo atual (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)***.
  - **Final do Jogo:** Verificação da situação de fim do jogo, com identificação do vencedor. O predicado deve chamar-se ***game_over(+GameState, -Winner)***.
  - **Lista de Jogadas Válidas:** Obtenção de lista com jogadas possíveis. O predicado deve chamar-se ***valid_moves(+GameState, -ListOfMoves)***.
  - **Avaliação do Estado do Jogo*:** Forma(s) de avaliação do estado do jogo do ponto de vista de um jogador, quantificada através do predicado ***value(+GameState, +Player, -Value)***.
  - **Jogada do Computador*:** Escolha da jogada a efetuar pelo computador, dependendo do nível de dificuldade, através de um predicado ***choose_move(+GameState, +Level, -Move)***. O nível 1 deverá devolver uma jogada válida aleatória. O nível 2 deverá devolver a melhor jogada no momento (algoritmo *míope*), 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), nível(eis) de dificuldade a usar no(s) jogador(es) artificial(ais), 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 intermédio 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**).

Sugere-se o seguinte escalonamento de tarefas, de acordo com o plano de aulas previsto:

- Semana de 6 de dezembro: escolha de grupo / tema;
- Semana de 13 de dezembro: escrita da secção de descrição do jogo;
- Semanas de 20 de dezembro, 27 de dezembro e 3 de janeiro: definição da representação interna do estado do jogo, implementação dos predicados de validação e execução de jogada, e de deteção de final de jogo;
- Semana de 10 de janeiro: implementação dos predicados de visualização do tabuleiro, menus de jogo e interação com o utilizador;
- Semana de 17 de janeiro: implementação do ciclo de jogo, ligando os predicados definidos anteriormente, e do jogador artificial – predicados de obtenção de jogadas válidas, de avaliação do tabuleiro, e obtenção de jogada do computador *.

***** Nota: a implementação do predicado de avaliação do tabuleiro e do nível 2 de dificuldade é opcional, sendo contabilizado como uma funcionalidade extra [valorização: até +1 valor].

Problemas (Jogos) propostos:

1. 4Mation - https://boardgamegeek.com/boardgame/329175/4mation
2. Breakthrough - https://boardgamegeek.com/boardgame/321224/breakthrough-tanks
3. Dag en Nacht - https://boardgamegeek.com/boardgame/347536/dag-en-nacht
4. Doblin - https://boardgamegeek.com/boardgame/308153/doblin
5. Five Field Kono - https://boardgamegeek.com/boardgame/25471/five-field-kono
6. Gi-Go - https://boardgamegeek.com/boardgame/348086/gi-go
7. Jeson Mor - https://boardgamegeek.com/boardgame/37917/jeson-mor
8. Jostle - http://www.marksteeregames.com/Jostle_Go_rules.pdf
9. Lines of Action - https://boardgamegeek.com/boardgame/3406/lines-action
10. Mitozo - https://boardgamegeek.com/boardgame/338168/mitozo
11. Pathway - http://www.marksteeregames.com/Pathway_rules.pdf
12. Petteia (Simple) - https://www.nestorgames.com/rulebooks/PETTEIA_EN.pdf
13. Quixo - https://boardgamegeek.com/boardgame/3190/quixo
14. Renpaarden - https://boardgamegeek.com/boardgame/70925/renpaarden
15. Shi - https://boardgamegeek.com/boardgame/319861/shi
16. Snort - https://boardgamegeek.com/boardgame/151888/snort
17. Splinter - https://splinterboardgame.blogspot.com/2021/06/splinter-is-two-player-abstractstrategy.html
18. Taktikus - https://boardgamegeek.com/boardgame/80811/taktikus
19. Zola - http://www.marksteeregames.com/Zola.pdf