Do Faculdade de Engenharia da Universidade do Porto

Mestrado Integrado em Engenharia Informática e Computação

Relatório Final

"Froglet"

Programação em Lógica (PLOG) – 2017/2018 – 1ºSemestre

Turma 1, Froglet3

Trabalho realizado por:

Diogo Luís Rey Torres	201506428	up201506428@fe.up.pt
Francisco Teixeira Lopes	201106912	ei11056@fe.up.pt

Resumo

O problema abordado neste relatório é a construção de um programa capaz de simular um jogo de tabuleiro. O problema foi resolvido usando Prolog como linguagem de programação, incluindo a lógica base do jogo como a respetiva visualização acompanhada de uma interface.

A lógica foi implementada com base em predicados lógicos consistentes e com variados propósitos, desde predicados auxiliares que fazem cálculos matemáticos, manipulação de listas a simples verificações. A representação do tabuleiro é feita com recurso a caracteres ASCII. E a interface foi concebida de modo a ser simples para o utilizador, sem no entanto ser incompleta.

Quanto à abordagem em específico, o problema foi resolvido de forma incremental. Começando pela representação do tabuleiro, manipulação do tabuleiro, construção do jogo nos diferentes modos. Após a realização destes passos foi implementado a regra dos saltos múltiplos, o score de cada jogador e a interface de interação com o utilizador.

Deste projeto resultou um programa capaz de jogar um jogo de tabuleiro conhecido por Froglet, o programa deixa jogar em modo de jogador contra jogador, em modo de jogador contra computador e ainda computador contra computador com diferentes níveis de dificuldade.

Com este projeto conclui-se ainda que é possível construir um jogo em Prolog e principalmente que esta linguagem permite abordar a inteligência artificial de uma maneira bastante simples. Permitiu também potenciar as nossas capacidades de raciocínio abstrato e de representação de problemas de forma declarativa ao familiarizar-nos com os paradigmas da Programação em Lógica.

Índice

1. Introdução

O trabalho foi desenvolvido no âmbito da unidade curricular de Programação Lógica. O objetivo do trabalho foi criar um programa, em Prolog, capaz de simular um jogo de tabuleiro.

A motivação do trabalho foi o jogo, aquando a escolha de tema, ter despertado o maior interesse.

O relatório descreve o jogo desenvolvido, as suas regras, a implementação em Prolog, dando atenção especial a certos detalhes da implementação, quando tal é merecido, e finalmente, uma breve conclusão.

2. O Jogo

2.1. História

Froglet é uma variante de um jogo de tabuleiro preservado pelo historiador de jogos de tabuleiro, Harold Murray. No seu livro, publicado em 1898, Murray descreve o jogo original e uma variante criada por si. A variante de Murray foi adaptada pela comunidade BrainKing para servir de versão online do jogo, esta é a variante denominada Froglet e a qual se vai implementar, as diferenças residem apenas no tamanho do tabuleiro e na distribuição de peças coloridas no tabuleiro.

2.2. Regras

O jogo consiste num tabuleiro 12x12, que é preenchido aleatoriamente com peças em forma de sapo. Estas peças podem ser de 4 cores e a distribuição final tem de ser: 66 verdes, 51 amarelos, 21 vermelhos e 6 azuis. O objetivo é capturar o número máximo de pontos e cada cor tem um valor diferente, verde vale 1, amarelo vale 2, vermelho vale 3 e azul vale 4.

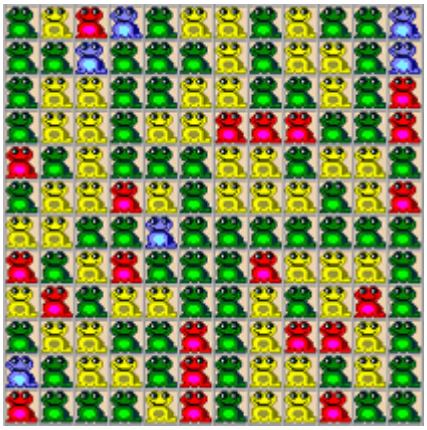

Figura 1 - Exemplo disposição inicial

O jogo começa com a remoção de qualquer sapo verde, após essa remoção, o jogo prossegue através de saltos. Para um salto ser válido, um jogador deve selecionar um sapo que tenha outro sapo diretamente adjacente, horizontalmente ou verticalmente, e que tenha um espaço vazio na direção do salto. Saltos múltiplos são permitidos mas não são obrigatórios, o jogador pode parar a qualquer salto sem ter de realizar os que se seguem. Porém, um jogador tem de efetuar no mínimo um salto.

Figura 2 - Exemplo salto, a vermelho sapo selecionado

Figura 3 - Depois do salto

O jogo termina quando não existirem mais saltos possíveis e o vencedor é quem tiver mais pontos.

3. A Lógica de Jogo

3.1. Representação do estado do jogo

O tabuleiro de jogo é representado através de uma lista de listas, para representar o tabuleiro utilizam-se valores numéricos que correspondem a pontuação de cada célula: 1 corresponde ao sapo verde, 2 ao sapo amarelo, 3 ao sapo vermelho, 4 ao sapo azul e por fim o 0 corresponde à célula vazia.

Exemplo de estado inicial, intermédio e final em Prolog:

```
initial([
        [2,2,2,1,1,1,1,1,1,1,2,2],
        [1,3,3,3,3,2,2,1,1,2,2,2],
        [1,1,2,3,1,1,1,2,1,1,2,4],
        [3,1,2,3,3,1,1,1,1,1,2,2],
        [1,1,2,2,3,1,1,1,1,1,2,2],
        [3,2,2,1,1,3,3,1,1,2,2,2],
        [1,1,2,2,1,2,1,1,2,1,2,2],
        [1,2,2,4,1,2,2,1,1,2,2,2],
        [3,2,2,1,1,1,3,4,2,1,2,2],
        [3,4,2,3,1,1,2,3,4,2,1,4],
        [1,1,1,3,1,3,1,3,2,1,1,2],
        [1,2,1,1,1,3,1,2,1,1,1,2]]).

ongoing([
        [2,2,2,1,1,1,1,1,1,1,2,2],
        [1,3,3,3,3,2,2,1,1,2,2,2],
        [1,1,2,0,1,1,1,0,0,0,0,0],
        [3,1,2,0,0,1,0,0,1,0,2,2],
        [1,1,0,0,0,0,0,0,0,0,2,2],
        [3,2,2,0,1,0,0,1,0,0,2,2],
        [1,1,2,2,1,0,1,1,0,1,2,2],
        [1,2,2,4,1,2,2,1,1,2,2,2],
        [3,2,2,1,1,1,3,4,2,1,2,2],
        [3,4,2,3,1,1,2,3,4,2,1,4],
        [1,1,1,3,1,3,1,3,2,1,1,2],
        [1,2,1,1,1,3,1,2,1,1,1,2]]).
```

```
final([
        [2,0,0,0,0,1,0,0,0,0,1,0],
        [0,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,1,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,3,0,0,0,0,0,0,2,0,2],
        [1,0,0,0,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,2],
        [1,0,0,1,0,0,0,0,0,0,0,0],
        [0,0,0,0,0,0,0,0,0,0,0,0],
        [0,2,0,0,1,0,2,0,0,4,0,2],
        [0,0,0,0,0,0,0,1,0,0,0,0],
        [0,0,0,0,0,0,4,0,1,0,0,0]]).
```

3.2. Visualização do tabuleiro em modo de texto

A representação interna do tabuleiro é diferente da representação em modo de texto. Assim os valores numéricos são convertidos para os símbolos GYRB. Porém, a representação em modo de texto é planeada de forma a ser mais agradável de visualizar que o formato denso de lista de listas.

O predicado de visualização é displayBoard(Board) e o output é o seguinte:

```
| ?- initial(X), displayBoard(X).
   A     B     C     D     E     F     G     H     I     J     K     L
|     |     |     |     |     |     |     |     |     |     |     |     |
-------------------------------------------------------------------------
|  Y  |  Y  |  Y  |  G  |  G  |  G  |  G  |  G  |  G  |  G  |  Y  |  Y  | 1
-------------------------------------------------------------------------
|  G  |  R  |  R  |  R  |  R  |  Y  |  Y  |  G  |  G  |  Y  |  Y  |  Y  | 2
-------------------------------------------------------------------------
|  G  |  G  |  Y  |  R  |  G  |  G  |  G  |  Y  |  G  |  G  |  Y  |  B  | 3
-------------------------------------------------------------------------
|  R  |  G  |  Y  |  R  |  R  |  G  |  G  |  G  |  G  |  G  |  Y  |  Y  | 4
-------------------------------------------------------------------------
|  G  |  G  |  Y  |  Y  |  R  |  G  |  G  |  G  |  G  |  G  |  Y  |  Y  | 5
-------------------------------------------------------------------------
|  R  |  Y  |  Y  |  G  |  G  |  R  |  R  |  G  |  G  |  Y  |  Y  |  Y  | 6
-------------------------------------------------------------------------
|  G  |  G  |  Y  |  Y  |  G  |  Y  |  G  |  G  |  Y  |  G  |  Y  |  Y  | 7
-------------------------------------------------------------------------
|  G  |  Y  |  Y  |  B  |  G  |  Y  |  Y  |  G  |  G  |  Y  |  Y  |  Y  | 8
-------------------------------------------------------------------------
|  R  |  Y  |  Y  |  G  |  G  |  G  |  R  |  B  |  Y  |  G  |  Y  |  Y  | 9
-------------------------------------------------------------------------
|  R  |  B  |  Y  |  R  |  G  |  G  |  Y  |  R  |  B  |  Y  |  G  |  B  | 10
-------------------------------------------------------------------------
|  G  |  G  |  G  |  R  |  G  |  R  |  G  |  R  |  Y  |  G  |  G  |  Y  | 11
-------------------------------------------------------------------------
|  G  |  Y  |  G  |  G  |  G  |  R  |  G  |  Y  |  G  |  G  |  G  |  Y  | 12
-------------------------------------------------------------------------
```

Figura 4 - Output atual e pretendido

3.3. Lista de Jogadas Válidas

A função validMoves e as chamadas das suas funções auxiliares são responsáveis pela obtenção de todos os movimentos possíveis no tabuleiro. Realizando uma pesquisa ao longo do tabuleiro, a partir de cada posição verifica as jogadas possíveis e agrega a uma lista de jogadas possíveis. No fim, retorna uma lista de listas com todas as jogadas possíveis no momento no formato [[P, Xi-Yi, Xf-Yf], […], …]. Em que a posição 0, corresponde à pontuação obtida na realização dessa jogada. Na posição 1, as coordenadas iniciais do salto e na posição 2 as coordenadas da posição final.

Para verificar se uma jogada é válida, o algoritmo verifica se a posição inicial e a posição final da jogada estão dentro dos limites do tabuleiro. Verifica também se a posição inicial é um sapo e se a posição final é uma célula vazia. Entre a posição inicial e final tem de existir um sapo e só se podem realizar os saltos em posições adjacentes a este sapo, sem contar com diagonais, em que a distância entre as posições seja igual a 2.

3.4. Execução de Jogadas

A primeira jogada a ser realizada é a remoção de um qualquer sapo verde por parte do primeiro jogador. A partir desse momento, as jogadas seguintes seguem o padrão normal.

Esse padrão consiste em: Selecionar e verificar o sapo a mover; Selecionar e validar a posição para a qual se irá mover o sapo; verificar se esse salto é válido; mover o sapo e obter o tabuleiro atualizado.

Assim, a escolha do sapo a ser movido necessita de cumprir certos requisitos para que essa seleção possa ser validada. É também importante referir que esta função de validar a jogada encontra-se no modo repeat até que o utilizador escolha posições válidas.

Existe ainda a possibilidade de o jogador realizar um salto múltiplo, isto é, se a partir da jogada que realizou ainda existirem mais jogadas válidas a partir do destino da jogada, o jogador pode optar por continuar a realizar saltos ou simplesmente terminar a jogada.

3.5. Avaliação do Tabuleiro

No jogo Froglet não existe uma avaliação do tabuleiro, mas sim a avaliação da jogada. Cada jogada tem uma pontuação associada, e quanto maior for essa pontuação melhor é a jogada.

A pontuação está relacionada com o valor do sapo que é "comido" durante a jogada, variando de 1 a 4 pontos.

3.6. Final do Jogo

O final do jogo é dado quando a função validMoves retornar uma lista com tamanho 0, isto é, quando não existir mais jogadas possíveis. Sendo, o vencedor do jogo aquele que obteve uma maior pontuação até ao momento.

3.7. Jogada do Computador

O tipo de jogada efetuada pelo computador varia consoante a dificuldade selecionada no inicio do jogo – easy ou hard.

A função cpuMove é responsável pela chamada de predicados que tratam da escolha da próxima jogada do computador, esta chamada varia consoante a dificuldade.

Em modo "easy" o procedimento de escolha é bastante direto, realizando uma chamada à função validMoves para obter uma lista com as possíveis jogadas e consequentemente é selecionado de maneira aleatória uma dessas jogadas para ser realizada pelo computador. Ainda permite ao cpu, decidir também de maneira aleatória se deve realizar saltos múltiplo ou não, caso a jogada que tenha feito o possibilite.

Por outro lado no modo "hard" apesar de o método ser bastante idêntico, a lista das jogadas possíveis é ordenada por ordem da pontuação e é escolhida a jogada com maior pontuação no momento. Não é realizada nenhuma verificação a partir daquelas jogadas, ou seja, o computador não prevê jogadas futuras nem saltos múltiplos. No entanto, se no caso da jogada com melhor pontuação permitir saltos múltiplos, o cpu realiza esse salto, escolhendo sempre o sapo mais valioso caso exista escolha no salto múltiplo.

4. Interface com o utilizador

Antes do jogo começar, o utilizador é questionado sobre o tipo de jogadores que irão jogar, humano ou computador. Esta escolha é feita para o jogador 1 e jogador 2.

Após esta escolha, e caso existam jogadores do tipo computador, o utilizador é questionado quanto à dificuldade do computador, a escolha é feita uma ou duas vezes ,dependendo do número de computadores em jogo.

O jogo inicia-se pela remoção de um sapo verde do tabuleiro, esta remoção é sempre feita pelo primeiro jogador e caso seja humano são pedidas as coordenadas. Caso contrário, é apenas mostrada a decisão do computador e o programa espera por input do utilizador para continuar.

A partir desta fase, o jogo processa-se sempre da mesma forma, sendo que ao jogador, são pedidas coordenadas de um sapo para realizar um salto, de seguida são pedidas coordenadas de destino do salto. Caso seja um salto válido o programa mostra o tabuleiro atualizado e prossegue ao próximo jogador em caso de não existirem saltos múltiplos possíveis a partir do destino desta última jogada. Se existirem saltos múltiplos é pedido ao utilizador coordenadas de destino do salto, apenas são necessárias estas pois as de origem serão as do último salto.

Se um ou ambos jogadores forem computador, é apresentado ao utilizador a jogada escolhida, e o programa só avança quando o utilizador confirmar.

No final do jogo é mostrado o resultado final e proclamado o vencedor.

5. Conclusões

Ao longo do projeto foi possível chegar a várias conclusões acerca de Prolog em contraste com programação imperativa. Usando esta linguagem, conseguimos perceber a facilidade que esta apresenta, por exemplo, na implementação de inteligência e de alguns raciocínios típicos do paradigma.

O feedback do trabalho é sobretudo positivo, precisamente pelo facto de nos ter permitido expandir o nosso conhecimento acerca desta linguagem, o que em futuros projetos nos permitirá utilizar esta linguagem para realizar cálculos relacionados com inteligência artificial.

O trabalho podia ser melhorado em três aspetos:

- Decisão de jogada do computador com previsão de estado futuro do tabuleiro
- Prioritização dos saltos múltiplos na decisão de jogada do computador
- Remoção de cortes vermelhos

Bibliografia

https://en.wikipedia.org/wiki/Leap_Frog_(board_game)

https://brainking.com/en/GameRules?tp=54