Spangles

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo 04:
Nuno Pinto - 201307878
Nuno Silva - 201200642

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

8 de Novembro de 2015

Resumo

Foi abordada o desenvolvimento, na linguagem PROLOG, o tabuleiro de jogo Spangles com o objetivo de aprofundar os conhecimentos da linguagem de programação assim como o desenvolvimento de código de forma declarativa.

A resolução do problema passou pelo desenvolvimento de vários predicados que permitissem a manipulação de listas de uma forma dinâmica, permitindo assim a criação de um tabuleiro com vários tamanhos possíveis, e predicados que permitissem aceder aos elementos destas listas para obter informações vitais para o funcionamento do jogo (jogadas disponíveis, a melhor jogada possível realizável, etc.).

A nível de resultados, foi possível a criação deste jogo em PROLOG e concluir que o planeamento de cada predicado de modo a ter em conta todos os possíveis casos de um predicado através da criação de grafos é essencial para a criação de qualquer programa em PROLOG. Para além do mais, a manipulação das várias listas presentes no programa e os diferentes elementos que podem constituir esta lista (outras listas e números) é de maior importância para a criação da larga maioria dos predicados envolvidos neste jogo de tabuleiro.

Conteúdo

1 Introdução

Os objetivos deste trabalho passavam pelo desenvolvimento de um jogo de tabuleiro, Spangles no caso do nosso grupo, assim como o aprofundamento da manipulação de listas e programação declarativa inevitavelmente presentes no desenvolvimento deste jogo de tabuleiro.

No relatório, é possível encontrar na secção seguinte uma breve descrição do jogo, a sua história e regras (O Jogo Spangles), uma explicação detalhada da lógica por detrás do jogo, a representação do estado do jogo (secção Lógica do Jogo, subsecção Representação do Estado do Jogo), os predicados envolvidos na visualização do tabuleiro (secção Lógica do Jogo, subsecção Visualização do Tabuleiro), a descrição do funcionamento da lista de jogadas válidas (secção Lógica do Jogo, subsecção Lista de Jogadas Válidas), os predicados envolvidos na execução de jogadas e como estas são realizadas (secção Lógica do Jogo, subsecção Execução de Jogadas), a descrição do predicado utilizado para a verificação do jogo e o seu funcionamento (secção Lógica do Jogo, subsecção Final do Jogo) assim como o predicado envolvido numa jogada do computador (secção Lógica do Jogo, subsecção Jogada do Computador). Por fim, é feita uma breve descrição da interface utilizada para interagir com o utilizador (secção Interface com o Utilizador) e as conclusões retiradas deste projecto (secção Conclusões).

2 O Jogo Spangles

O jogo Spangles foi publicado pela primeira vez em 1982 pela companhia nestorgames, tendo sido desenhado por David L. Smith com a parte artística desenvolvida por Néstor Andrés. Este jogo utiliza peças de duas cores diferentes, ambas com forma triangular, e cada jogador tem como objetivo criar um triângulo maior constituído por quatro triângulos em que cada canto deste triângulo é uma peça colocada no tabuleiro por um único jogador (a peça no centro deste triângulo não é relevante) assim como impedir que o jogador adversário faça ele próprio um triângulo maior constituído por quatro triângulos. Fora a peça inicial, cada uma das peças jogadas pelo jogador só pode ser colocada no tabuleiro ao lado de outra peça, independentemente da cor ou de o triângulo estar invertido ou na forma normal.

3 Lógica do Jogo

3.1 Representação do Estado do Jogo

O Spangles representa o estado do seu tabuleiro com uma lista de listas, sendo esta dinamicamente aumentada à medida que são introduzidos mais triângulos no tabuleiro. Cada lista de listas em si pode conter até 25 elementos (visto que a cada jogador estão associados, no máximo, 25 triângulos), sendo estes elementos valores inteiros numerados de 0 a 5. Estes valores inteiros representam vários tipos de triângulos:

- 0 representa um triângulo um triângulo na forma normal, inacessível a qualquer jogador, composto por espaços brancos e cuja nalidade consiste em evitar a desformatação do tabuleiro.
- 1 representa um triângulo invertido, inacessível a qualquer jogador, composto por espaços brancos e cuja nalidade consiste em evitar a desfor-

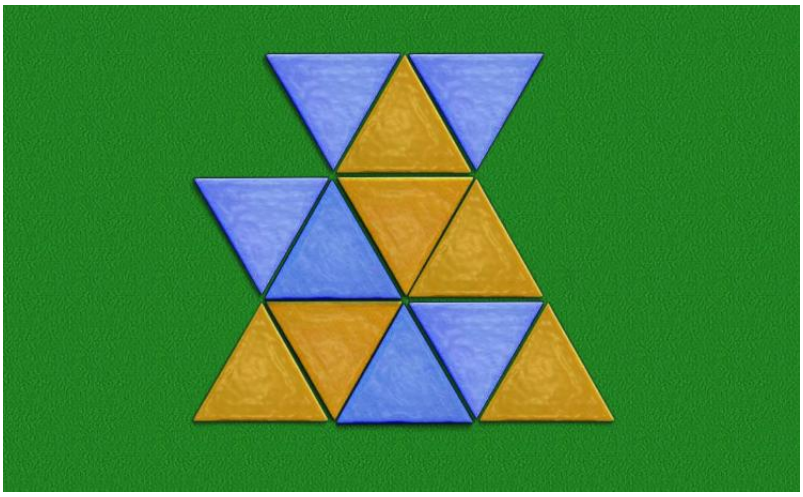

Figura 1: Estado intermédio do jogo, com peças azuis a serem controladas pelo jogador 1 e castanhas pelo jogador 2.

matação do tabuleiro.

- 2 representa um triângulo na forma normal colocado no tabuleiro pelo jogador 1.
- 3 representa um triângulo invertido colocado no tabuleiro pelo jogador 1.
- 4 representa um triângulo na forma normal colocado no tabuleiro pelo jogador 2.
- 5 representa um triângulo invertido colocado no tabuleiro pelo jogador 2.

No estado inicial do jogo, encontra-se apenas um triângulo no tabuleiro, jogado por qualquer um dos jogadores e representado em Prolog por [[2]] ou [[4]] (o triângulo inicial colocado no tabuleiro não pode ser um triângulo invertido). Durante o decorrer do jogo, irão ser colocados vários triângulos no tabuleiro, jogados por ambos os jogadores e representados em Prolog por uma lista do género correspondente à gura 1 (peças azuis do jogador 1, castanhas do jogador 2): [[0, 3, 4, 3], [3, 2, 5, 4], [4, 5, 2, 3, 5]]). O estado nal do jogo ocorreria quando um dos jogadores formasse um triângulo maior constituído por quatro triângulos, com três dos triângulos que colocou no tabuleiro a compôr as pontas deste triângulo maior (o triângulo no centro do triângulo maior pode ser colocado por qualquer um dos jogadores), representado em Prolog por uma lista do género correspondente à gura 2 (peças brancas são do jogador 1, vermelhas do jogador 2) em que o jogador 1 venceria: ([[0, 1, 2, 1, 4], [1, 2, 5, 4, 3, 2], [4, 3, 2, 5, 2, 5, 2], [1, 0, 5]).

3.2 Visualização do Tabuleiro

No jogo são utilizados vários predicados para a visualização do tabuleiro em modo de texto. O primeiro predicado é o printB([H/T]) em que [H/T] é uma lista de listas a ser visualizada na consola. Este é um predicado com recursividade na cauda em que são escolhidas entre as várias opções:

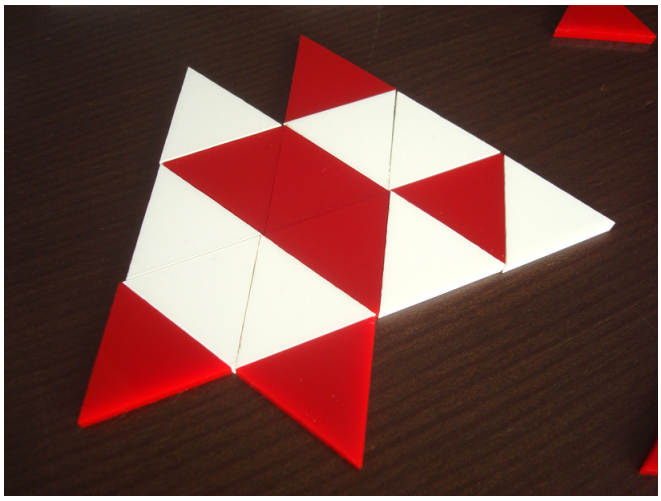

Figura 2: Estado nal do jogo, com peças brancas a serem controladas pelo jogador 1 e vermelhas pelo jogador 2.

- Se o primeiro triângulo da lista de listas for um triângulo com forma normal, é chamado o predicado printB norm([H/T]) que coloca o espaçamento inicial na consola correto para a visualização do tabuleiro.
- Se o primeiro triângulo da lista de listas for um triângulo com forma invertida, é chamado o predicado printB inv([H/T]) que coloca o espaçamento inicial na consola correto para a visualização do tabuleiro.

Dentro de tanto o printB norm e printB inv (ambos predicados com recursividade na cauda) estão quatro outros predicados. Os triângulos estão divididos em quatro partes (início, primeira parte do meio, segunda parte do meio e nal) e cada predicado é responsável pelo desenho de uma parte:

- O predicado print line rst([H/T]), em que [H/T] é uma lista de valores inteiros, desenha o topo dos triângulos com forma normal e as bases dos triângulos invertidos na lista.
- O predicado print line sec(X), em que [H/T] é uma lista de valores inteiros, desenha a primeira parte do meio dos triângulos com forma normal e a segunda parte do meio do triângulos invertidos na lista.
- O predicado print line third(X), em que [H/T] é uma lista de valores inteiros, desenha a segunda parte do meio dos triângulos com forma normal e a primeira parte do meio dos triângulos invertidos.
- O predicado print line last(X), em que [H/T] é uma lista de valores inteiros, desenha a base dos triângulos com forma normal e o topo dos triângulos invertidos.

Estes predicados usam quatro outros predicados, twrite rst(X), twrite sec(X), twrite third(X), twrite last(X), em que X é um valor inteiro que representa o tipo de triângulo, responsáveis por desenhar as várias partes dos triângulos individualmente (portanto print line rst([H/T]) chama twrite rst(X) sucessivamente até a primeira linha ser desenhada, print line sec([H/T]) faz o mesmo com twrite sec(X), etc.).

3.3 Lista de Jogadas Válidas

O predicado available_plays(+IteratedBoard, +ControlBoard, +Column, +Row, +ListLength, -ListofPlays, +TempListofPlays, +Iterator) recebe duas versões do tabuleiro, um tabuleiro que vai ser iterado durante a execução do predicado e outro que vai irá ser utilizado para a procura de elementos no tabuleiro, a coluna e linha em que estão a ser verificadas possíveis jogadas, o tamanho da linha atual (tamanho de Row), uma lista que vai ser iterada durante a execução do predicado e, no final da execução do predicado, guardada na variável ListofPlays assim como um iterador.

O predicado funciona através da procura de seis casos possíveis:

- Na posição [linha, coluna] são analisadas quaisquer jogadas possíveis no topo do triângulo caso este seja um triângulo voltado para baixo colocado por um jogador no tabuleiro (3 ou 5). É acrescentada a jogada [linha-1, coluna] à lista de jogadas possíveis caso no topo deste triângulo não se encontre nenhum triângulo controlado por um jogador.

- Na posição [linha, coluna] são analisadas quaisquer jogadas possíveis em [linha, coluna-1] caso o triângulo nesta posição seja um triângulo controlado pelo jogador (2, 3, 4 ou 5). É acrescentada a jogada [linha, coluna-1] à lista de jogadas possíveis caso no lado esquerdo deste triângulo, na mesma linha, não se encontre nenhum triângulo controlado por um jogador.

- Na posição [linha, coluna] são analisadas quaisquer jogadas possíveis em [linha, coluna+1] caso o triângulo nesta posição seja um triângulo controlado pelo jogador (2, 3, 4 ou 5). É acrescentada a jogada [linha, coluna+1] à lista de jogadas possíveis caso no lado direito deste triângulo, na mesma linha, não se encontre nenhum triângulo controlado por um jogador.

- Na posição [linha, coluna] são analisadas quaisquer jogadas possíveis debaixo do triângulo caso este seja um triângulo em forma normal colocado por um jogador no tabuleiro (2 ou 4). É acrescentada a jogada [linha+1, coluna] à lista de jogadas possíveis caso debaixo deste triângulo não se encontre nenhum triângulo controlado por um jogador.

- Os últimos dois casos são casos em que é sempre possível acrescentar uma jogada à lista de jogadas: É sempre possível jogar um triângulo no lado esquerdo do primeiro triângulo controlado por um jogador numa linha assim como jogar um triãngulo no lado direito do útlimo triângulo controlado por um jogador numa linha.

O tamanho da linha, neste predicado, serve para indicar a mudança de linha no predicado. Caso o valor da coluna exceda o tamanho da lista, então já foram encontradas todas as jogadas possíveis nessa linha. Como tal, o predicado irá procurar jogadas possíveis na próxima linha.

O iterador serve para garantir a presença de várias jogadas numa posição. Caso um dos casos acima suceda, o predicado é voltado a ser chamado com a adição da nova jogada à lista temporária de jogadas e com o iterador correspondente ao caso. Assim, quando o predicado é voltado a ser chamado na mesma posição [linha, coluna], o iterador irá indicar o caso em que sucedeu e só irá verificar os casos em que Icaso é maior que o Iactual, permitindo assim guardar várias jogadas na mesma posição.

3.4 Execução de Jogadas

O predicado verif_play(+AvailablePlays, +Play) garante a validação de uma jogada num tabuleiro por parte do jogador. O predicado make play lê, caso um jogador humano esteja ativo, a linha e coluna em que o jogador está a tentar fazer uma jogada. O verif play devolve true or false dependendo se a jogada [linha, coluna] faça parte das jogadas disponíveis e, como tal, seja uma jogada legal.

O predicado make_playaux(+Board, +ActivePlayer, +Row, +Column, -NewBoard) permite a execução de uma jogada num tabuleiro, podendo esta jogada cair em cinco casos diferentes:

- Caso o jogador esteja a tentar jogar um triângulo numa nova linha inicial, é chamado o predicado make_playaux_row(+Board, +ActivePlayer, +Row, +Column, -NewBoard, +NewRow, +NewRowTemp) em que é feito o append de uma nova linha ao tabuleiro no início do tabuleiro. Caso o novo triângulo a ser acrescentado não seja inserido na primeira coluna da nova primeira linha a ser criada, é chamado o predicado make_playaux_rowinc(+Column, -NewRow, +NewRowTemp, +Counter) que preenche a nova linha com triângulos vazios (0 e 1) até à coluna em que foi acrescentada o novo triângulo controlado pelo jogador para evitar a desformatação do tabuleiro.

- Caso o jogador esteja a tentar jogar um triângulo numa nova linha final, é chamado o predicado make_playaux_row(+Board, +ActivePlayer, +Row, +Column, -NewBoard, +NewRow, +NewRowTemp) em que é feito o append de uma nova linha ao tabuleiro no final do tabuleiro. Caso o novo triângulo a ser acrescentado não seja inserido na primeira coluna da nova última linha a ser criada, é chamado o predicado make_playaux_rowinc(+Column, -NewRow, +NewRowTemp, +Counter) que preenche a nova linha com triângulos vazios (0 e 1) até à coluna em que foi acrescentada o novo triângulo controlado pelo jogador para evitar a desformatação do tabuleiro.

- Caso o jogador esteja a tentar jogar um triângulo numa nova coluna inicial, é chamado o predicado make_playaux_column(+Board, +ActivePlayer, +Row, +Column, -NewBoard, +HeadofBoard) em que é feito o append de um novo triângulo ao início da linha em que é acrescentada a peça. De modo a evitar a desformatação das coordenadas do tabuleiro com a presença da nova peça na primeira coluna (o que leva à criação de uma nova coluna no tabuleiro), é acrescentada a todas as linhas do tabuleiro um triângulo vazio para compensar a criação da nova coluna. Isto é feito através do predicado update_rboard(+Board, -NewBoard, +NewBoardTemp).

- Caso o jogador esteja a tentar jogar um triângulo numa nova coluna final, é chamado o predicado make_playaux_column(+Board, +ActivePlayer, +Row, +Column, -NewBoard, +HeadofBoard) em que é feito o append de um novo triângulo ao final da linha em que é acrescentada a peça.

- Caso o jogador esteja a tentar jogar um triângulo numa posição ocupada por um triângulo invisível, é utilizado o predicado replace(+Board, +Row, +Column, +Element, -NewBoard) para substituir o triângulo invisível por uma peça jogada por um jogador.

3.5 Final do Jogo

O predicado verif_solution(+IterativeBoard, +ActivePlayer1, +ActivePlayer2, +ControlBoard, +Row, +Column, +ListLength, +AP1Triangles, +AP2Triangles, +Difficulty) permite a verificação para a possibilidade de uma solução ter sido encontrada após a realização de uma jogada. O predicado irá iterar sobre todos os elementos do tabuleiro de modo a verificar os dois casos possíveis:

- Caso o elemento a ser iterado seja neste momento um triângulo voltado para cima (2 ou 4) irá verificar se, na coluna anterior e seguinte, na linha abaixo, se encontram dois triângulos do mesmo jogador. Caso esse caso se verifique, o jogador ativo conseguiu formar um triângulo de dimensão superior igual a 4 e, como tal, vence o jogo.
- Caso o elemento a ser iterado seja neste momento um triângulo voltado para baixo (3 ou 5) irá verificar se, na coluna anterior e seguinte, na linha acima, se encontram dois triângulos do mesmo jogador. Caso esse caso se verifique, o jogador ativo conseguiu formar um triângulo de dimensão superior igual a 4 e, como tal, vence o jogo.

Caso não seja encontrada uma solução, o predicado reduz o número de peças do jogador ativo e chama o make_play do jogador em espera. Caso o número de peças de cada jogador seja igual a 0, o jogo acaba num empate.

3.6 Jogada do Computador

O predicado find_bestplay(+Board, +ActivePC, +AvailablePlays, -BestPlayList, +TempBestPlayList, +Points) permite determinar o valor de cada uma das jogadas disponíveis na lista de jogadas disponíveis. São dados os seguintes pontos a uma jogada:

- São dados o número máximo de pontos (5) caso a jogada disponível atual seja uma jogada vencedora.
- São dados 4 pontos à jogada disponível atual caso esta impeça o outro jogador de vencer.
- São dados 2 pontos à jogada disponível caso esta prepare uma jogada vencedora na próxima ronda (ou seja, o computador vai favorecer colocar uma peça a duas casas de uma outra peça que já colocou na mesma linha para possivelmente, na próxima jogada, poder colocar uma peça vencedora entre as duas na linha superior ou inferior).
- São dados o menor número de pontos (1) a qualquer outra jogada.

Após a determinação do valor de cada jogada, caso a dificuldade do PC esteja no modo difícil (1), o make_play do PC irá procurar a jogada de maior valor na lista com a ajuda do predicado max e irá realizar essa jogada. Caso o PC esteja no modo médio (0), o make_play do PC irá escolher uma jogada aleatória de entre as jogadas disponíveis.

4 Interface com o Utilizador

A interface com o utilizador é realizada com o predicado read. O jogador tem a possibilidade de escolher o modo de jogo inicial, a dificuldade do PC no modo de jogo contra o PC assim como a coluna e linha onde deseja jogar.

5 Conclusões

É possível concluir, neste projecto, que a preparação prévia à elaboração de cada predicado é vital. Sem esta preparação, é muito fácil tentar realizar cada predicado como se se tratasse de uma função de qualquer linguagem de programação, um erro crasso que muitas vezes resulta em predicados mal elaborados e pouco eficientes. Saber manipular listas através dos diferentes predicados associadas a estas acabou também por ser essencial, assim como a elaboração de uma preparação para cada caso que se pode encontrar na realização de um predicado.

A nível de melhorias no trabalho, além de um modo de jogo não-funcional (o modo de PC contra PC), pensamos que uma redução de alguns dos maiores predicados em predicados menores poderia ter resultado num programa mais eficiente assim como fácil de ler.

Bibliografia

https://boardgamegeek.com/boardgame/37103/spangles

http://stackoverflow.com/