Dominup em Prolog

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Dominup4:
Ângela Filipa Pereira Cardoso - up200204375
Nuno Miguel Rainho Valente - up200204376

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

8 de Novembro de 2015

Resumo

O trabalho por nós desenvolvido teve como principal objetivo a recriação de um jogo, o Dominup. Este é um jogo de tabuleiro sendo uma variação do famoso jogo de Dominó, aparentemente surgido na China há mais de dois mil anos.

O problema consisitiu em desenvolver o jogo em PROLOG SICSTUS e apresentar a sua interação com o utilizador na forma mais prática possível.

O Dominup será, posteriormente, alvo de maior detalhe neste relatório no que diz respeito a toda a sua jogabilidade, contudo, globalmente, permitimos que o jogo fosse jogado de três formas distintas:

- Humano contra Humano.
- Humano contra Computador.
- Computador contra Computador.

Nos tipos de jogo que envolvem o computador foi necessário recorrer à implementação de inteligências artificiais, duas neste caso - uma mais simples e uma outra mais complexa.

Conteúdo

1 Introdução

O relatório está organizado por capítulos, sendo o primeiro correspondente a esta introdução. No segundo capítulo fazemos uma apresentação detalhada do jogo Dominup com imagens ilustrativas que facilitam o entendimento do jogo. Pretendemos dar a conhecer a lógica de jogo e a sua forma de implementação no jogo em Prolog. Toda esta informação está presente no capítulo três. Indicamos como funciona o módulo de interface com o utilizador em modo de texto - capítulo quatro, e fazemos alguns reparos finais nas conclusões - capítulo cinco. Terminamos com a bibliografia que utilizamos e em anexo apresentamos todo o código fonte do projeto.

2 O Jogo Dominup

Dominup é uma variação do jogo Dominó para 2 a 4 jogadores, em que, tal como o nome sugere, é possível colocar peças em cima de outras.

No típico Dominó existem 28 peças duplas numeradas de 0 a 6, à semelhança das faces de um dado. Já no Dominup há 36 peças duplas numeradas de 0 a 7, usando códigos binários: o ponto no centro representa 1, o circulo pequeno representa 2 e o circulo grande representa 4, como se pode ver na figura 1. Este desenho das peças, juntamente com as regras do Dominup e de dois outros jogos, foram criadas por Néstor Romeral Andrés em 2014, sendo o conjunto publicado por nestorgames[1].

Figura 1: Exemplo da peça $3 \cdot 6$.

Existem dois tipos de colocação de peças no Dominup:

- subir - a peça é colocada em cima de duas peças adjacentes que estejam ao mesmo nível, de forma a que os números da peça colocada sejam iguais aos que ficam por baixo (um em cada peça de suporte), tal como mostra a figura 2.
- expandir - a peça é colocada na superfície de jogo, de forma a que fique adjacente e ortogonal a pelo menos uma peça já colocada, como, por exemplo, as duas peças já colocadas na figura 2.

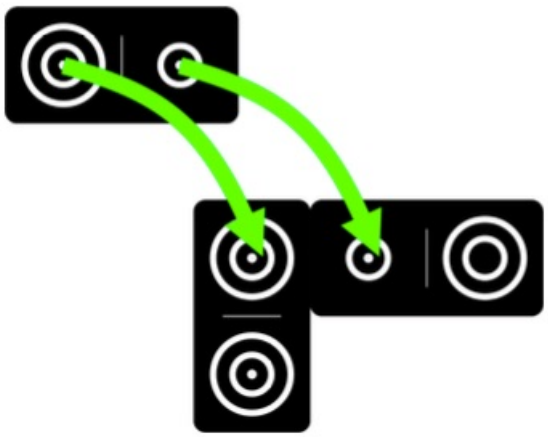

Figura 2: Exemplo de um posicionamento a subir válido.

[1] http://www.nestorgames.com

Tal como no Dominó, as regras são relativamente simples. Começa-se por distribuir as peças aleatoriamente e de forma equilibrada pelos jogadores, mantendo a face voltada para baixo.

O jogador com o duplo 7 inicia o jogo, colocando essa peça no centro da superfície de jogo e determinando a ordem dos restantes jogadores, que é dada pelo sentido contrário ao ponteiro dos relógios.

Começando no segundo, cada jogador, na sua vez, realiza ambos os passos seguintes:

1. Enquanto for possível, coloca peças a subir, podendo escolher a ordem em que o faz;
2. Se ainda tiver alguma peça, coloca-a a expandir.

Se, no final da sua vez, o jogador ficar sem peças, é declarado vencedor e o jogo termina. Alternativamente, os restantes jogadores podem continuar, de forma a determinar o segundo, terceiro e quarto lugares.

Na figura 3 pode ser observado um possível jogo de Dominup a decorrer.

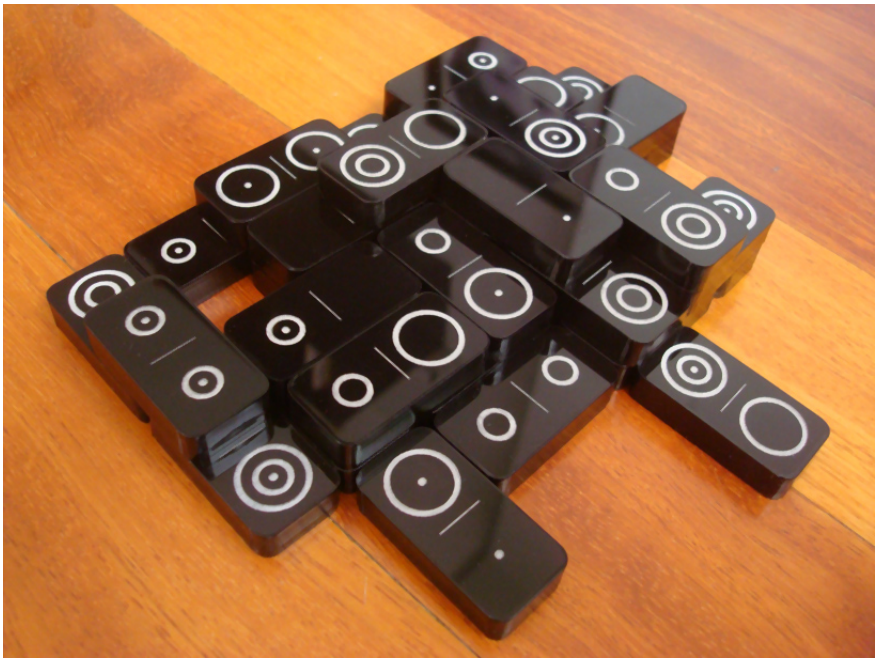

Figura 3: Exemplo de um jogo de Dominup.

3 Lógica do Jogo

Pretendemos nos subcapítulos que se seguem descrever o projeto e implementação da lógica do jogo em Prolog, incluindo a forma de representação do estado do tabuleiro e sua visualização, a execução de movimentos, verificação do cumprimento das regras do jogo, determinação do final do jogo e cálculo das jogadas a realizar pelo computador utilizando diversos níveis de jogo.

3.1 Representação do Estado do Jogo

O tabuleiro de jogo no Dominup é um quadriculado, cujo tamanho não deve limitar o posicionamento de peças a expandir. Uma vez que expansões sucessivas têm que ser feitas ortogonalmente, uma linha expansiva numa só direção ocupa $2 + 1 + 2 + 1 + \cdots$ quadrículas, assumindo que a primeira peça está colocada horizontalmente. Ao todo temos 36 peças, por isso seriam necessárias $2*18+18 = 54$ quadrículas para acomodar uma tal linha de expansão. Dado que a primeira peça é colocada no centro do tabuleiro, se a expansão fosse feita sempre no mesmo sentido, poderíamos ter que considerar um tabuleiro com 108 quadrículas de lado.

Uma análise mais cuidada das regras do jogo revela que as peças são preferencialmente colocadas a subir. De facto, em cada vez, um jogador coloca tantas peças a subir quanto possível e no máximo

uma peça a expandir. Além disso, em geral não será boa estratégia para nenhum dos jogadores expandir sempre no mesmo sentido. Tendo em consideração as limitações de um computador, quer em termos de capacidade de processamento, quer em termos de tamanho do ecrã, decidimos considerar um tabuleiro quadriculado de lado 24. Desta forma, é possível colocar pelo menos 8 peças em cada um dos 4 sentidos a partir do centro do tabuleiro. Eventualmente, com a experiência, podemos decidir reduzir ou aumentar este tamanho. O ideal é que o tabuleiro nunca limite as jogadas, mantendo-se tão pequeno quanto possível para facilitar o processamento.

Sendo assim, o tabuleiro em Prolog é representado por uma lista `board` de 24 linhas do tabuleiro. Por sua vez, cada linha é uma lista de 24 elementos, um para cada quadrícula. Cada elemento de uma linha é um termo complexo `halfPiece(Number, Level, Cardinal)` representando a meia peça de dominó lá colocada, com o seguinte significado:

- `Number` é o número da meia peça;
- `Level` é o nível do tabuleiro em que a peça está colocada, 1 se for colocada em cima do tabuleiro, 2 se for colocada em cima dessa, etc;
- `Cardinal` é o ponto cardeal que indica a posição da outra metade da peça.

Assim, numa quadrícula vazia temos `halfPiece(_, _, _)`. Quando é colocado o dominó duplo 7, teremos `halfPiece(7, 1, e)` e `halfPiece(7, 1, o)` em duas quadrículas adjacentes. Numa fase mais avançada pode ser colocado o dominó $2 \cdot 6$ no nível 3 na vertical com `halfPiece(2, 3, n)` e `halfPiece(6, 3, s)`.

Além do tabuleiro, o estado de jogo contém as peças de cada jogador. Dado que apenas consideramos dois jogadores nesta implementação, teremos as listas `set1` e `set2` com as peças dos jogadores 1 e 2, respetivamente. À medida que vão sendo colocadas no tabuleiro, as peças são retiradas destas listas. Cada peça é representada pelo termo complexo `piece(Number1, Number2)`, por exemplo, `piece(2, 6)`, convencionando-se que o primeiro número é sempre menor ou igual ao segundo.

Finalmente, o estado de jogo terá indicação de qual é o próximo jogador a jogar na variável `playerTurn`, que pode ser 1 ou 2. Assim, o estado de jogo é representado por `state(set1, set2, playerTurn, board)`.

Um possível estado inicial, já com as peças distribuídas aleatoriamente pelos jogadores, é dado por:

- ```
  set1([piece(0, 1), piece(0, 3), piece(0, 4), piece(0, 6), piece(1, 1),
        piece(1, 2), piece(1, 4), piece(1, 5), piece(1, 7), piece(2, 2),
        piece(2, 3), piece(2, 6), piece(3, 3), piece(3, 7), piece(4, 5),
        piece(5, 6), piece(6, 7), piece(7, 7)]);
  ```
- ```
  set2([piece(0, 0), piece(0, 2), piece(0, 5), piece(0, 7), piece(1, 3),
        piece(1, 6), piece(2, 4), piece(2, 5), piece(2, 7), piece(3, 4),
        piece(3, 5), piece(3, 6), piece(4, 4), piece(4, 6), piece(4, 7),
        piece(5, 5), piece(5, 7), piece(6, 6)]);
  ```
- ```
  playerTurn(1);
  ```
- ```
  board([[halfPiece(_, _, _), halfPiece(_, _, _), ..., halfPiece(_, _, _)], ...,
         [halfPiece(_, _, _), halfPiece(_, _, _), ..., halfPiece(_, _, _)]]).
  ```

Após algumas jogadas, podemos chegar ao estado ilustrado pela Figura 4. Em cima estão as peças do jogador 1, dado que esse é o próximo a jogar. Em baixo está o tabuleiro com as peças que contém até então. Cada meia peça colocada no tabuleiro tem do lado esquerdo o seu número e do lado direito o nível em que se encontra. Este estado intermédio é dado pelos seguintes termos:

- ```
  set1([piece(0, 1), piece(0, 3), piece(0, 4), piece(0, 6), piece(1, 2),
        piece(1, 4), piece(1, 5), piece(1, 7), piece(2, 2), piece(2, 6),
        piece(3, 7), piece(4, 5), piece(5, 6), piece(6, 7)]);
  ```
- ```
  set2([piece(0, 0), piece(0, 2), piece(0, 7), piece(1, 3), piece(1, 6),
        piece(2, 7), piece(3, 4), piece(3, 6), piece(4, 4), piece(4, 6),
        piece(5, 5), piece(5, 7)]);
  ```
- ```
  playerTurn(1);
  ```
  ```

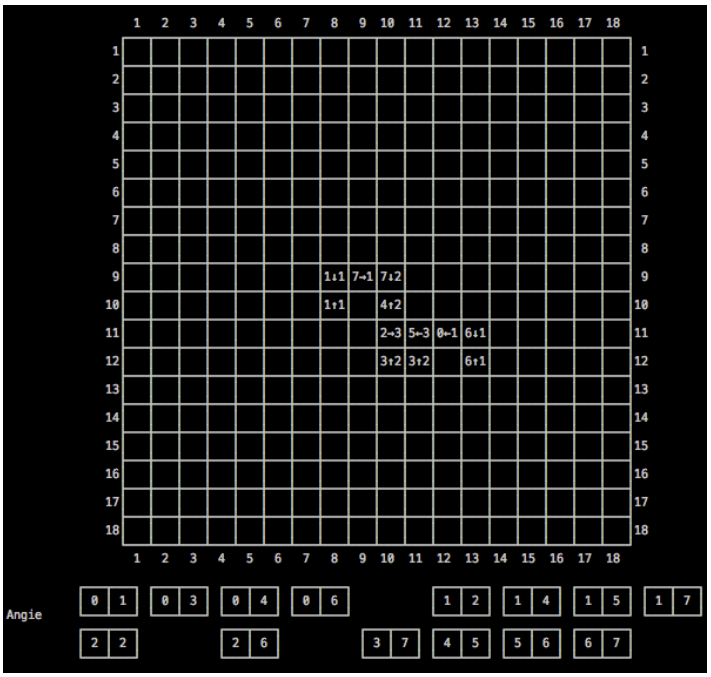

Figura 4: Estado intermédio do jogo.

- `board([[halfPiece(_, _, _), halfPiece(_, _, _), ..., halfPiece(_, _, _)],`
  `...,`
  `[halfPiece(_, _, _), halfPiece(_, _, _), ..., halfPiece(_, _, _)],`
  `[..., halfPiece(1, 1, s), halfPiece(7, 1, e), halfPiece(7, 2, s), ...]`
  `[..., halfPiece(1, 1, n), halfPiece(_, _, _), halfPiece(4, 2, n), ...]`
  `[..., halfPiece(2, 3, e), halfPiece(5, 3, o), halfPiece(0, 1, o),`
  `halfPiece(6, 1, s), ...]`
  `[..., halfPiece(3, 2, n), halfPiece(3, 2, n), halfPiece(_, _, _),`
  `halfPiece(6, 1, n), ...]`
  `[halfPiece(_, _, _), halfPiece(_, _, _), ..., halfPiece(_, _, _)],`
  `...,`
  `[halfPiece(_, _, _), halfPiece(_, _, _), ..., halfPiece(_, _, _)]]).`

Um possível estado final, correspondente ao jogo da Figura 3, deverá ter a ilustração da Figura 5 e, à exceção do tabuleiro, cuja representação é demasiado complexa para traduzir neste relatório, tem os seguintes termos:

- `set1([]);`
- `set2([]);`
- `playerTurn(2);`

3.2 Visualização do Tabuleiro

Para visualizar o tabuleiro em modo de texto, começamos por idealizar o formato das Figuras 4 e 5. No entanto, para tal seria necessário desenhar quadrículas diferentes consoante as posições das peças, o que é relativamente complexo de executar. Sendo assim, pelo menos para já, cada quadrícula será sempre desenhada com traços a toda a volta, independentemente da posição das peças. Cada peça é colocada em duas quadrículas adjacentes e a relação entre as duas metades é identificada pelo conteúdo das células. Isto é, o dominó $2 \cdot 5$ dá origem às células `23e` e `53o`, se for colocado no nível 3 na horizontal com o 2 à esquerda do 5.

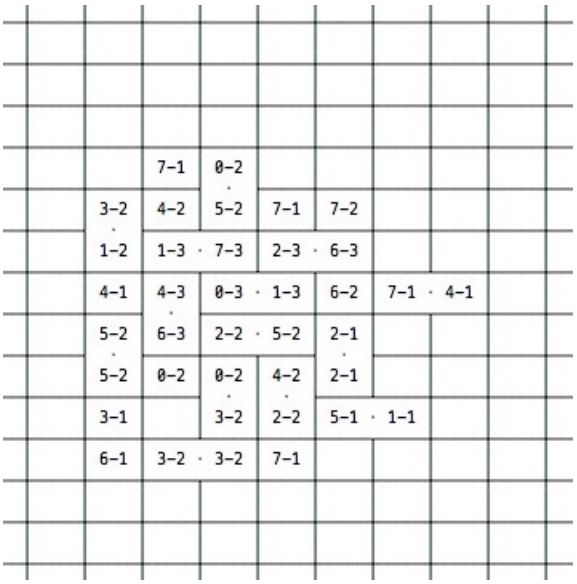

Figura 5: Estado final do jogo.

Para obter a visualização descrita acima, usamos os seguintes predicados em Prolog:

- `display_game([X|Y]) :- nl, putLetters, nl,`
  `write('  ----------------------------------------'), nl,`

  `rows(1, [X|Y]), putLetters, nl, nl,!.`

- `rows(N, [X|Y]):-putLeftNumbers(N), analyse(X), putRightNumbers(N), nl,`
  `putGrid(N), N1 is N+1, nl, rows(N1, Y).`
  `rows(_,[]).`

- `putLetters:-write('    A   B   C   D   E   F   G   H   I   J   K   ... ').`

- `putLeftNumbers(N):-((N<10, write(0), write(N), write('')) ;` (write(N), write('—'))).—

- `putRightNumbers(N):-((N<10, write(0), write(N)) ; (write(N))).`

- `putGrid(N):-((N<24, write(' |---|---|---|---|---|---|---|---|---|---|---|---|');`
  `write('  ----------------------------------------')) ).`

- `analyse([]):-!.`

  `analyse([X|Z]):-print(X),  write('|'), analyse(Z).`

De facto, os predicados são ligeiramente diferentes, por o LaTeXnão aceita os caracteres unicode usados para desenhar as quadrículas. De qualquer forma, o resultado está disponível na Figura 6 cuja configuração é a mesma da Figura 4.

3.3 Lista de Jogadas Válidas

Obtenção de uma lista de jogadas possíveis. Exemplo: *valid_moves(+Board, -ListOfMoves).*

3.4 Execução de Jogadas

Validação e execução de uma jogada num tabuleiro, obtendo o novo estado do jogo. Exemplo: *move(+Move, +Board, -NewBoard).*

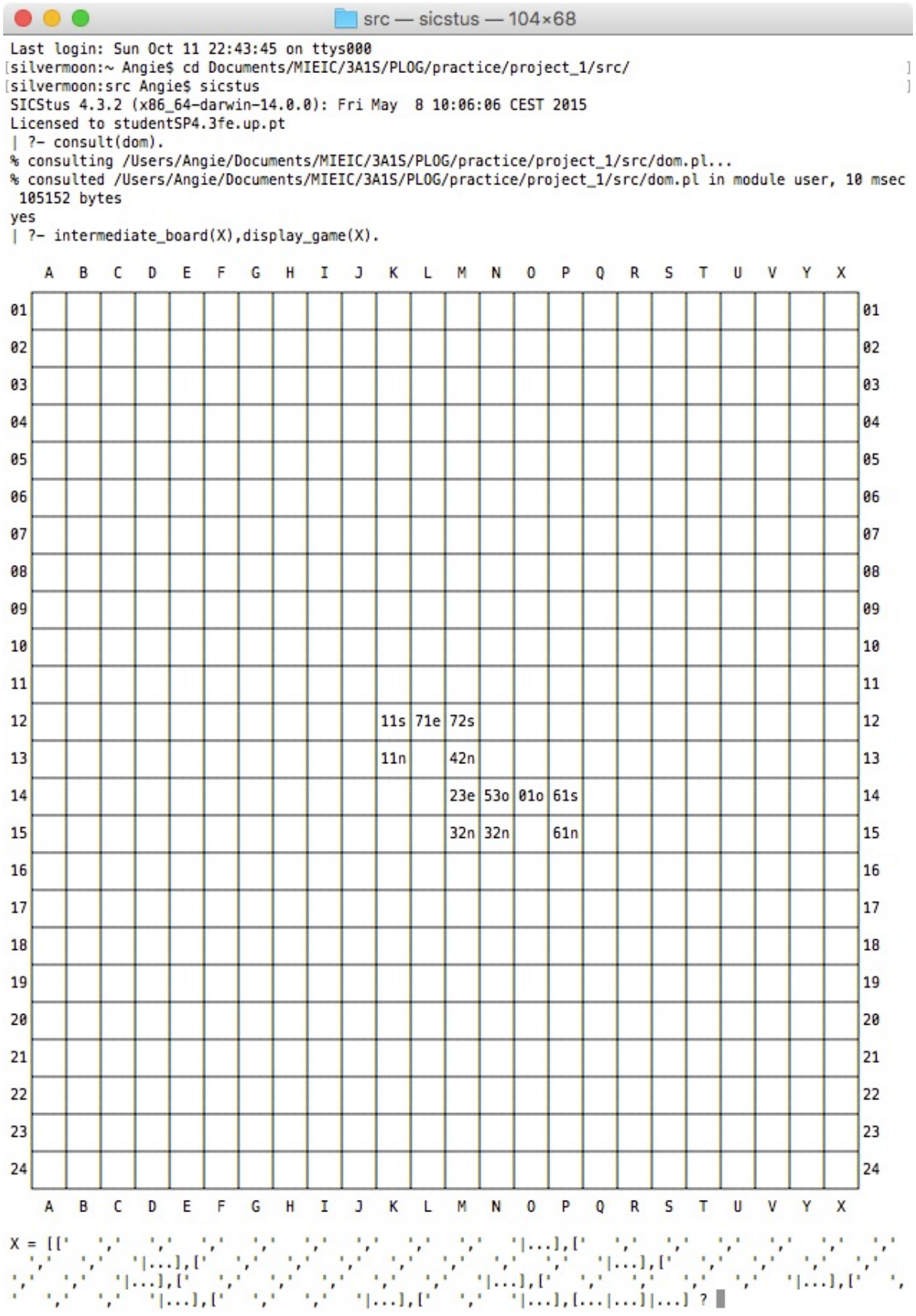

Figura 6: Visualização do tabuleiro em Prolog.

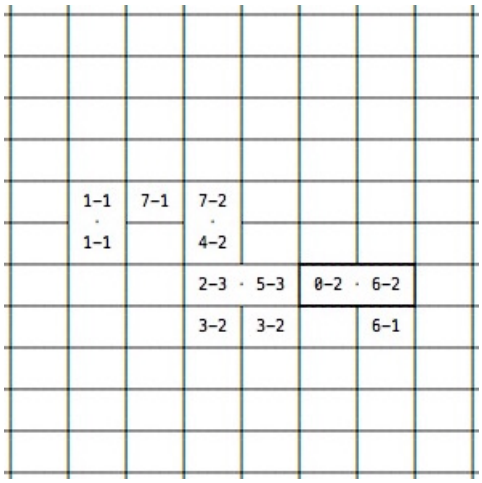

Figura 7: Movimento a subir.

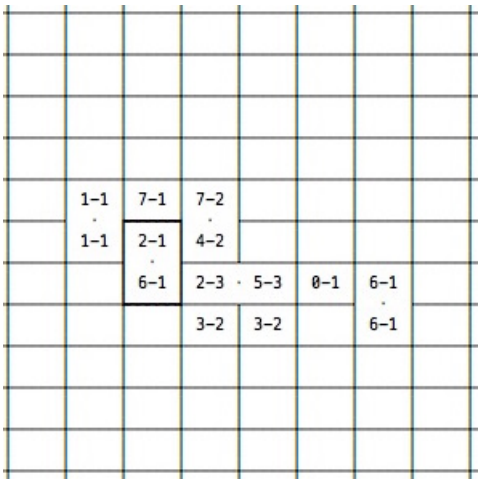

Figura 8: Movimento a expandir.

3.5 Avaliação do Tabuleiro

Avaliação do estado do jogo, que permitirá comparar a aplicação das diversas jogadas disponíveis. Exemplo: *value(+Board, +Player, -Value).*

3.6 Final do Jogo

Verificação do fim do jogo, com identificação do vencedor através de *checkGameOver.*

3.7 Jogada do Computador

Escolha da jogada a efetuar pelo computador, dependendo do nível de dificuldade *choose_move(+Level, +Board, -Move).*

- uma mais simples que de entre as jogadas válidas seleciona uma qualquer, e uma outra mais complexa que avalia para cada jogada possível qual a mais eficaz para levar o computador a vencer.

4 Interface com o Utilizador

O utilizador apenas necessita de ir digitando alguns carateres no teclado, seguidos da tecla Enter, interagindo com o jogo. No ecrã inicial o jogador tem de escolher se pretende iniciar um novo jogo ou um previamente gravado. Se seguir um jogo já gravado, este é carregado com as definições inicialmente selecionadas pelo utilizador e o jogo continuará a se desenrolar. Caso contrário, o jogador tem de escolher se pretende jogar contra o computador, contra outro jogador ou então observar um jogo entre computador vs computador. O nível de dificuldade, fácil ou difícil será escolhido logo a seguir. Para terminar a configuração do jogo falta apenas escrever o(s) nome(s) do(s) jogador(es).

O jogo inicia mostrando o tabuleiro de jogo com a peça 7 - 7 colocada no meio do tabuleiro. A quem pertencesse esta peça no início do jogo tem direito a continuar a jogar. E para jogar, cada jogador tem acesso às suas peças e é convidado a escolher o seguinte para efetivar uma jogada:

- colocar o número da peça do lado esquerdo que pretende jogar
- colocar o número da peça do lado direito que pretende jogar
- indicar onde pretende colocar a metade esquerda da peça, indicando para isso a linha e a coluna
- indicar a orientação da peça, dizendo para isso onde se situa a metade direita da peça em relação à metade esquerda, utilizando as opções "n", "e", "s", "w", designando norte, este, sul, oeste, respetivamente.

Todas as outras navegações durante o jogo referidas, à exceção da última, será necessário introduzir um número para avançar na configuração da jogada.

5 Conclusões

Este jogo foi desde logo uma das nossas primeiras hipóteses de seleção, de entre quatro possíveis, para se desenvolver o primeiro projeto em Prolog que seria objeto de avaliação. Ficamos extremamente felizes com o desenvolvimento de um jogo que à partida trazia alguma nostalgia pelo simples facto de nos lembrar alguns momentos, em que outrora um simples jogo de Dominó fazia a delícia dos mais pequenos.

Somos da opinião que este jogo trás uma espécie de reformulação do típico jogo de dominó e que faz todo o sentido coloca-lo disponível a jogar num computador, apesar de ainda em modo de texto.

Relativamente ao projeto consideramos que tudo o que delineamos e que todos os pontos exigidos na implementação do jogo foram cumpridos. Foi muito proveitoso uma vez que serviu também para praticarmos e aumentarmos a nossa prática em programação lógica.

Gostaríamos de ter desenvolvido mais um nível de inteligência artificial e ter colocado a dimensão do tabuleiro de jogo a ser adaptado dinamicamente, isto é, consoante a colocação de peças e os movimentos de expand,o tabuleiro ir aumentando de dimensão.

Todavia, estamos agradados com o rumo que o projeto tomou e o resultado final foi o que esperavamos.

Bibliografia

- http://www.nestorgames.com
- https://sicstus.sics.se/sicstus/docs/4.0.3/html/sicstus/Saving.html
- https://sicstus.sics.se/sicstus/docs/latest4/pdf/sicstus.pdf
- Apontamentos das aulas teóricas

A Código fonte do jogo Dominup

Apresenta-se a seguir todo o código fonte utilizado no projeto: