Epaminondas

Relatório Intercalar

Universidade do Porto

Faculdade de Engenharia

FEUP

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo 13:
Hugo Ari Rodrigues Drumond - 201102900
João Alexandre Gonçalinho Loureiro - 200806067

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

13 de Outubro de 2013

Resumo

O jogo de tabuleiro que iremos desenvolver chama-se Epaminondas. As peças presentes no tabuleiro são todas iguais e movem-se como o Rei no xadrez, exceto quando se agrupam numa linha, coluna ou diagonal. Trata-se do 1º. trabalho de grupo, da disciplina de Programação em Lógica, realizado pelo Grupo 13.

1 Introdução

O fim deste trabalho é criar um jogo de tabuleiro com base na matéria exposta nas aulas teóricas e téorico-práticas. Este trabalho é interessante porque obriga-nos a pensar de um ponto de vista puramente lógico, isto é, cria-se uma base de dados lógica através de cláusulas, sem controlo de fluxo (explícito), e depois procuramos por possíveis soluções. Este paradigma de programação chama-se programação declarativa.

2 O Jogo Epaminondas

Este jogo, inicialmente chamado *Crossings* e jogado num tabuleiro 8x8, foi descrito pela primeira vez no Livro *A Gamut of Games* em 1969. Depois da publicação deste livro, Bob Abbott, reconfigurou o tabuleiro de 8x8 para 14x12, de modo a tornar o jogo nas diagonais mais importante, e renomeou o nome para Epaminondas como homenagem ao general TheBan que inventou a phalanx, uma formação de guerra usada em 371 a.c para derrotar o exército espartano.[1, 2]

Neste jogo, são posicionadas 28 peças iguais em cada lado maior do tabuleiro, com cores diferentes. São as brancas a iniciar o jogo, e depois joga-se alternadamente. Os movimentos de cada peça são iguais ao do Rei do xadrez, pode andar uma casa em qualquer direção. Podem ser criados inúmeros grupos de peças, phalanxes, estas só se podem deslocar um número igual ou menor ao número de peças do grupo, na direcção da linha formada pelo grupo. Uma peça pode fazer parte de várias phalanxes. Uma phalanx pode ser dividida, mas só pode andar um número de casas igual ou menor ao numero de elementos da phalanx que se irá mover. Não é permitido passar um jogada, em cada jogada é obrigatório mover uma phalanx ou uma peça. Todas as peças têm de estar contidas no tabuleiro. Não podem haver peças sobrepostas ou na mesma posição, exceto quando a cabeça de uma phalanx captura outra peça ou phalanx. Para capturar é necessário atacar com um grupo com mais elementos que o adversário na linha de ataque. A peça/phalanx capturada é removida do tabuleiro para sempre. Um jogador ganha quando, na sua vez de jogar, tiver mais peças que o adversário na linha mais afastada de si.[2]

3 Representação do Estado do Jogo

Iremos criar um estado inicial, uma lista de listas da forma:

([[1,1,0,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,0,2,2],

[1,1,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,2,2]]).

Através da regra de interface play(XS,YS,XD,YD) iremos chamar as size e atualizarestado, que testam os limites do jogo e atualizam o estado, caso tal respeite as regras do jogo. A regra atualizarestado tenta unificar canimovethere, mudarestado e turnchange. Se a regra canimovethere não corresponder isso implica o falhanço do "if"em prolog e portanto nada acontece. Se canimovethere retornar true então podemos proceder à alteração do estado através da regra mudarestado. Restando mudar a vez do jogador, através de por exemplo:

turnchange(C) :- (C = white ->retract(turn(white)) ; asserta(turn(black))), (C = black ->retract(turn(black)) ; asserta(turn(white))).

O estado vai sendo atualizado consoante as jogadas de cada jogador, o 0 significa que não há nenhuma peça nessa posição. O 1 representa uma peça branca e o 2 uma preta.

Um estado com posições intermédias, ([[1,1,1,1,0,0,0,0,2,2,2,2], [0,0,0,0,0,0,0,0,0,0,0,0], [1,0,1,0,0,0,0,0,2,2,0,0], [1,1,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,2,2], [1,1,0,0,0,0,0,0,0,0,2,2]]).

Um estado com posições finais, qualquer configuração em que existam mais peças do adversário na minha linha mais recuada, na vez dele jogar. E viceversa.

4 Visualização do Tabuleiro

De modo a imprimir o estado atual do tabuleiro, tem-se de construir um predicado recursivo que escreva o conteúdo de cada linha na consola e depois dê um break line. Tal pode ser feito através das seguintes cláusulas:

```
14 %Através de duas cláusulas recursivas imprimir o estado atual do jogo
15 sepforprint( [] ).
16 sepforprint( [H|T] ):-
17   printline(H),
18   nl,
19   sepforprint(T).
20
21 printline([]).
22 printline([H|T]):-
23   write(H),
24   printline(T).
```

Resultando:

```
110000000022
110000000022
110000000022
110000000022
110000000022
110000000022
110000000022
110000000022
110000000022
110000000022
110000000022
110000000022
110000000022
110000000022

yes
| ?-
```

5 Movimentos

Cada peça pode ser movida sozinha ou em conjunto. No caso de ser só uma peça é possivel movê-la para qualquer direção, horizontal, vertical ou diagonal, em uma unidade desde que essa célula esteja vazia. No caso de se mover um conjunto de peças, estas não podem estar separadas por nenhuma célula vazia nem por peças do adversário, todas elas serão movidas até um máximo correspondente ao número de peças que se pretende mover e têm que respeitar a orientação na qual o grupo se encontra, por exemplo, se o grupo de peças se encontrar na horizontal as peças só podem ser movidas ou para a esquerda ou para a direita.

Como já foi referido iremos ter um predicado chamado canimovethere, que irá receber a peça, ou conjunto de peças, que se pretende mover e para onde queremos mover e que irá verificar se é possivel essa jogada.

6 Conclusões e Perspectivas de Desenvolvimento

O Epaminondas é um jogo cheio de estratégia e regras, que irão ser a parte mais difícil de implementar. A movimentação das peças que pode ser feita em oito direções e usando só uma peça ou um conjunto de peças, no caso dos phalanxs, sempre tendo em atenção os limites do tabuleiro e as outras peças, quer sejam do próprio jogador o que impossibilita a jogada ou do jogador adversário cujas peças têm de ser removidas do tabuleiro, no caso de ter menos peças no phalanx, ou então serão as peças do próprio a serem removidas, irá ser a parte mais desafiante a implementar.

O trabalho vai ser desenvolvido pelos dois elementos do grupo, com divisão de tarefas para que a quantidade de trabalho seja equivalente. Para nos ajudar a organizar e manter o nosso código usamos um sistema de controlo de versões, o git.

Estimamos que ainda falte cerca de 70(%) do trabalho, pois temos muitas das ideias já formuladas mas falta a sua implementação.

Bibliografia

[1] Bob Abbott. Epaminondas. http://www.logicmazes.com/games/epam.html, Revised: December 11, 2010. The website of the game creator.

[2] Kerry Handscomb. Abstract games issue 3 autumn 2000. http://www.logicmazes.com/games/epam/index.html, 3 Autumn 2000. Epaminondasm... a game of classical elegance.

A Nome do Anexo

Código Prolog implementado (representação do estado, cabeçalhos dos predicados para as jogadas e predicado que permite a visualização em modo de texto do tabuleiro).