Breakthrough

Relatório Intercalar

Universidade do Porto

Faculdade de Engenharia

FEUP

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo 17:
João Guedes - 070509043
João Henriques - 110509026

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

Setembro de 2011

Resumo

O *Breakthrough* é um jogo de tabuleiro tradicional semelhante às damas, premiado e com um conjunto de regras muito simples e acessível.

A implementação pretendida irá oferecer três modos de jogo - Humano/Humano, Humano/Computador e Computador/Computador - sendo que o Computador terá vários níveis de dificuldade.

Recorreremos à linguagem de programação *ProLog* para o fazer, e a *C++* para o módulo de visualização gráfico do jogo.

Conteúdo

1 Introdução

Este trabalho tem como principal motivação a consolidação do nosso conhecimento em ProLog, visto que o paradigma de programação é totalmente diferente daquele a que temos vindo a estar habituados ao longo do curso.

Por ser menos exigente e mais exequível dados os nossos conhecimentos, começaremos por implementar o modo Humano/Humano através de regras lógicas.

Seguidamente recorreremos a algoritmos de inteligência artificial e à teoria dos jogos para implementar os modos que envolvam o Computador.

Escolhemos este trabalho porque gostamos da sua simplicidade e fluidez, e apesar de não ter um conjunto de regras muito vasto, é um jogo com potencial estratégico. Se o oponente não estiver atento é capaz de ser vencido muito rapidamente.

De notar que, apesar da nossa implementação consistir num tabuleiro convencional de 8x8, o *Breakthrough* pode ser jogado com outras configurações, como tabuleiros 7x7 ou tabuleiros não-quadrados.

2 Descrição do Problema

O *Breakthrough* foi criado no ano 2000 por *Dan Troyka* e disponibilizado para a plataforma comercial *Zillion of Games*.

2.1 Preparação de jogo

1. Cada jogador começa com 16 peças uniformes, distribuidas pelas duas linhas mais próximas de si, à semelhança dum tabuleiro de xadrez.

2. Os dois jogadores escolhem o seu lado e é sorteado o primeiro a começar.

2.2 Movimentos permitidos

1. Um jogador pode mover uma peça se a casa de destino estiver vazia, diagonalmente ou em frente, conforme ilustrado.

2. Os jogadores movimentam-se sempre em direção à base do adversário, e nunca podem retroceder.

3. Quando o jogador avança em direção à base oposta e se depara com uma peça adversária diagonalmente, pode capturá-la, sendo a peça capturada eliminada do tabuleiro e substituída pela peça do capturador. No entanto a captura não é obrigatória nem encadeada como nas damas.

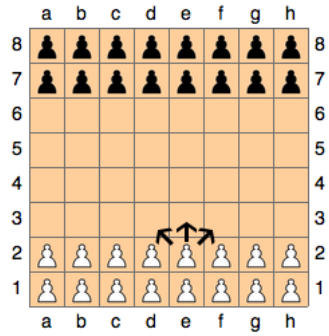
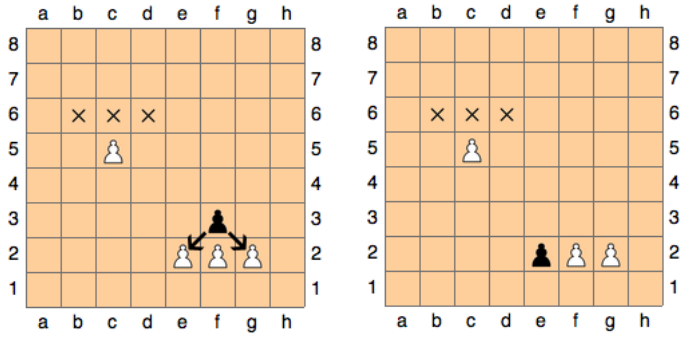


Figura 1: Movimento possível da peça e2



(a) Possibilidades para a peça f3 (b) Captura de e2

Figura 2: Processo de captura.

2.3 Vitória

1. O primeiro jogador a atingir a base do adversário, vence. No caso anterior, se o jogador 2 (peças pretas) atingisse a linha 1, venceria.
2. Se todas as peças de um jogador forem capturadas, este perde o jogo.
3. Um empate é matemáticamente impossível, e todas as peças têm sempre pelo menos uma jogada na diagonal possível.

3 Representação do Estado do Jogo

A representação do tabuleiro 8x8 é feita em Prolog por uma lista de listas. Cada item do tabuleiro é representado no formato "Jogador", em que este pode ser:

- 1 - jogador de peças brancas (Norte)
- 2 - jogador de peças pretas (Sul)
- 0 - casa vazia

Código fonte 1 Representação de tabuleiro inicial.

```
initBoard(
        [
                [ 1, 1, 1, 1, 1, 1, 1, 1 ],
                [ 1, 1, 1, 1, 1, 1, 1, 1 ],
                [ 0, 0, 0, 0, 0, 0, 0, 0 ],
                [ 0, 0, 0, 0, 0, 0, 0, 0 ],
                [ 0, 0, 0, 0, 0, 0, 0, 0 ],
                [ 0, 0, 0, 0, 0, 0, 0, 0 ],
                [ 2, 2, 2, 2, 2, 2, 2, 2 ],
                [ 2, 2, 2, 2, 2, 2, 2, 2 ]
        ]).
```

Código fonte 2 Representação de tabuleiro intermédio.

```
intermediumBoard(
        [
                [ 1, 1, 1, 1, 1, 1, 1, 1 ],
                [ 1, 1, 0, 1, 1, 1, 0, 1 ],
                [ 0, 0, 1, 0, 0, 0, 0, 0 ],
                [ 0, 0, 0, 0, 0, 1, 0, 0 ],
                [ 0, 0, 0, 0, 0, 2, 0, 0 ],
                [ 0, 0, 0, 2, 0, 0, 0, 2 ],
                [ 2, 2, 2, 0, 2, 0, 0, 2 ],
                [ 2, 2, 2, 2, 2 ,2 ,2, 2 ]
        ]).
```

Código fonte 3 Representação de tabuleiro final, em que o jogador 1 venceu.

```
finalBoard(
        [
                [ 1, 1, 1, 1, 1, 1, 1, 1 ],
                [ 1, 1, 0, 1, 1, 0, 0, 1 ],
                [ 0, 0, 1, 0, 0, 0, 0, 0 ],
                [ 0, 0, 0, 0, 0, 0, 0, 0 ],
                [ 0, 2, 0, 0, 0, 1, 0, 0 ],
                [ 0, 0, 0, 2, 0, 0, 0, 2 ],
                [ 2, 2, 0, 0, 2, 0, 0, 2 ],
                [ 2, 2, 2, 1, 2, 2, 2, 2 ]
        ]).
```

4 Representação de um Movimento

A descrição seguinte será feita tomando o jogador 2 (que avança na direcção Sul-Norte) como referência.

O jogador pode mover uma dada peça sua para uma casa adjacente a:

1. Norte, se esta estiver vazia.
2. Noroeste ou Nordeste, se esta estiver vazia ou contiver uma peça do oponente, sendo que neste último caso há forçosamente uma captura.

Para efetuar um movimento, chamamos a seguinte função:

Código fonte 4 Predicado para mover a peça.

```
movePawn([Ox,Oy], [Dx,Dy]).
```

Esta função aceita o X e Y da casa de origem e da casa de destino, e terá de verificar se ambas as casas são adjacentes, bem como validar a jogada, consoante o jogador que estiver na casa de origem.

Código fonte 5 Exemplificação de um movimento.

```
movePawn([ 7, 5 ], [ 6 , 4 ]).
```

A peça que se encontra em (7, 5), é movida para a casa (6, 4).

Para capturar uma peça do adversário na casa de destino, será usado o seguinte predicado:

Código fonte 6 Captura de peça na casa de destino.

```
capturePawn([Dx, Dy]).
```

5 Visualização do Tabuleiro

O tabuleiro do jogo, é apresentado no ecrã através do predicado "printBoard". Esta função apresenta as células do tabuleiro representadas pelo número do jogador correspondente, ou em branco, caso a mesma se encontre vazia. Apresenta também as linhas e colunas à volta do tabuleiro, numeradas do número 1 até ao número da ordem de colunas ou linhas.

Na construção do código para esta função, "printBoard", foram criados os seguintes predicados:

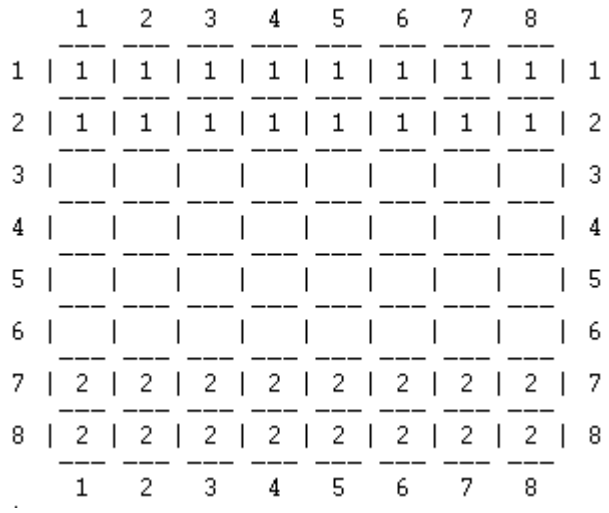


Figura 3: Tabuleiro impresso com "printBoard', mostrando o estado inicial de um jogo'.

Código fonte 7 Predicado "writePlayer' .

```
% Imprime a peça do jogador correspondente.
writePlayer(1) :-
        write('1').
writePlayer(2) :-
        write('2').
writePlayer(0) :-
        write(' ').
```

Código fonte 8 Predicado "printRow" .

```
% Critério de paragem.
printRow([]).

% Imprime a borda esquerda da célula, e a respetiva
% peça do jogador (cabeça da lista).
% Por fim chama recursivamente a mesma função com
% a cauda da lista, até encontrar o critério de paragem.
printRow([H|T]) :-
        write('| '),
        writePlayer(H),
        write(' '),
        printRow(T).
```

Código fonte 9 Predicado "printHorizSep" e "printColNumbers".

```
% Imprime a linha de separação horizontal.
printHorizSep   :-
        write('   --- --- --- --- --- --- --- ---').

% Imprime os números das colunas.
printColNumbers :-
        write('    1   2   3   4   5   6   7   8').
```

Código fonte 10 Predicado "printFullRow".

```
% Critério de paragem.
printFullRow([], _).

% Imprime o número da linha no ínicio e no fim de cada iteração,
% bem como o separador horizontal. Chama também o predicado que irá
% imprimir todas as células de cada linha, com a respectiva peça.
% Por fim chama recursivamente a mesma função com a cauda da
% lista (contendo as restantes células da linha correspondente),
% e o número actual da linha, até atingir o critério de paragem.
printFullRow([H|T], N) :-
        N1 is N+1,
        printHorizSep,
        nl,
        write(N),
        write(' '),
        printRow(H),
        write('| '),
        write(N),
        nl,
        printFullRow(T, N1).
```

Código fonte 11 Predicado "printBoard".

```
% Regra a ser usada quando é passado um tabuleiro vazio.
printBoard([]).

% Imprime o tabuleiro, chamando o predicado que imprime o número
% das colunas (no início e no fim do tabuleiro),
% e de seguida o predicado que imprime cada linha individual.
% É impresso também o separador horizontal do tabuleiro.
printBoard([H|T]) :-
        printColNumbers,
        nl,
        printFullRow([H|T], 1),
        printHorizSep,
        nl,
        printColNumbers.
```

6 Conclusões e Perspectivas de Desenvolvimento

Dados os conhecimentos adquiridos, podemos concluir que o código dos predicados para efetuar quer um movimento, quer uma captura, deverá ser simples de implementar. O trabalho já começou a ser desenvolvido em Prolog, e posteriormente será criado um interface em ambiente gráfico no contexto de LAIG.

Por enquanto, a implementação incidirá apenas no ambiente em modo de texto.

Estimamos que cerca de 15% do total do projeto esteja concluído, uma vez que os modos de jogo Humano/Computador e Computador/Computador, sendo baseadas na implementação de algoritmos de inteligência artificial, irão necessitar de mais investigação, e por conseguinte mais tempo dispendido.

Olhando otimisticamente para o projeto, achamos que a sua relativa simplicidade nos permitirá desenvolver a lógica de jogo suficientemente depressa para nos podermos concentrar em pequenos detalhes que tornarão o jogo o mais interessante possível.

Bibliografia

[1] Tutorial de prolog. http://hilltop.bradley.edu/~chris/prolog.html.

[2] Tutorial sobre bibliografias em prolog. http://en.wikibooks.org/wiki/LaTeX/Bibliography_Management.

[3] Dan Troyka. http://brainking.com/en/GameRules?tp=84, 2000.

A Código Prolog

Código fonte 12 Código dos predicados utilizados.

```
initBoard([[ 1, 1, 1, 1, 1, 1, 1, 1 ],
           [ 1, 1, 1, 1, 1, 1, 1, 1 ],
           [ 0, 0, 0, 0, 0, 0, 0, 0 ],
           [ 0, 0, 0, 0, 0, 0, 0, 0 ],
           [ 0, 0, 0, 0, 0, 0, 0, 0 ],
           [ 0, 0, 0, 0, 0, 0, 0, 0 ],
           [ 2, 2, 2, 2, 2, 2, 2, 2 ],
           [ 2, 2, 2, 2, 2, 2, 2, 2 ]]).
writePlayer(1) :-
        write('1').
writePlayer(2) :-
        write('2').
writePlayer(0) :-
        write(' ').
printRow([]).
printRow([H|T]) :-
        write('| '),
        writePlayer(H),
        write(' '),
        printRow(T).
printHorizSep   :-
        write('   --- --- --- --- --- --- --- ---').
printColNumbers :-
        write('    1   2   3   4   5   6   7   8').
printFullRow([], _).
printFullRow([H|T], N) :-
        N1 is N+1,
        printHorizSep,
        nl,
        write(N),
        write(' '),
        printRow(H),
        write('| '),
        write(N),
        nl,
        printFullRow(T, N1).
printBoard([]).
printBoard([H|T]) :-
        printColNumbers,
        nl,
        printFullRow([H|T], 1),
        printHorizSep,
        nl,
        printColNumbers.
init :-
        initBoard(A),
        printBoard(A).
% Protótipos do predicados da função de movimento e captura de uma peça.
% movePawn([Ox, Oy], [Dx, D2]).
% capturePawn([Dx, Dy]).
```