Jogo Cage

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo 2:
José Peixoto - 200603103
Luís Cruz - 201303248

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

13 de Novembro de 2016

Resumo

Resumo sucinto do trabalho com 150 a 250 palavras (problema abordado, objetivo, como foi o problema resolvido/abordado, principais resultados e conclusões).

No âmbito da unidade curricular de Programação em Lógica, foi-nos proposto o desenvolvimento de um jogo de tabuleiro em *Prolog*: o *Cage*. O principal objetivo na realização deste projeto foi a aquisição de novas competências na expressão de conceitos lógicos em *Prolog*.

Findo o projeto, realçamos a expressividade do *Prolog* para conceitos lógicos, e a nossa carência de experiência com este paradigma de programação.

Conteúdo

1 Introdução

Descrever os objetivos e motivação do trabalho. Descrever num parágrafo breve a estrutura do relatório.

2 O Jogo Cage

O Cage é um jogo de estratégia em tabuleiro semelhante às damas que foi inventado por Mark Steere em maio de 2010. O autor descreve-o como um jogo para dois jogadores sem qualquer informação oculta. É um jogo abstrato sem fator de sorte nem empates. É jogado num tabuleiro de damas 10x10 ou 8x8 e, ao contrário do jogo original das damas, todo tabuleiro está preenchido, no início, com peças já promovidas a "damas". "Jogo de aniquilação de alta energia" é a frase escolhida pelo autor para caricaturar o jogo, uma vez que o movimento para o centro do tabuleiro assegura a aniquilação, de pelo menos, uma das cores.

2.1 Regras

O Cage é jogado por dois jogadores num tabuleiro de damas com 50 damas vermelhas e 50 damas azuis na versão de tabuleiro 10x10 ou com 32 damas vermelhas e 32 damas azuis na versão de 8x8 tabuleiro. O tabuleiro é iniciado preenchendo todas as casas com damas de cor alternada.

2.1.1 Objetivo

Para vencer é necessário capturar todas as damas inimigas. No final, pode ganhar-se mesmo que se perca a última peça que se está a movimentar (saltar) para capturar todas as damas inimigas ainda em jogo.

2.1.2 Movimentos

Existem quatro tipos de movimentos:

1. Restrito
2. Centralizador
3. Adjacente
4. Salto

Durante um turno, um jogador apenas pode utilizar um tipo de movimento.

Restrição 1 Nunca se pode colocar uma dama ortogonalmente (horizontal ou verticalmente) adjacente a uma dama de cor idêntica. Nem de forma transitória durante um turno de vários movimentos.

Restrição 2 Nunca se pode movimentar uma dama que tenha adjacências ortogonais com damas inimigas para uma casa onde tal não aconteça.

Centralizador Este movimento de uma casa, permite à dama deslocar-se na horizontal, vertical ou diagonal para uma casa vazia e que permite que a dama se aproxime do centro do tabuleiro.

Adjacente Uma dama que não tenha adjacências ortogonais com damas inimigas pode mover-se apenas uma casa em qualquer direção que contenha adjacências ortogonais com uma ou mais damas inimigas.

Salto O movimento de salto permite capturar uma dama inimiga, movimentando a dama do jogador de uma casa ortogonalmente adjacente de um lado da dama inimiga para a casa vazia adjacente do lado oposto. É possível capturar uma dama inimiga nas casas periféricas do tabuleiro de uma casa adjacente e do lado oposto da dama inimiga na borda do tabuleiro. O resultado é que quer a dama capturada quer a dama que captura são removidas do tabuleiro.

3 Lógica do Jogo

No primeiro contato que o utilizador tem com o programa é-lhe solicitado o modo de jogo que permite, estando à disposição três distintos: humano contra humano, humano contra computador e computador contra computador.

```
| ?- cage.
        Cage game

  [1] Human vs. Human
  [2] Human vs. Computer
  [3] Computer vs. Computer

Enter game mode number:
|:
```

Figura 1: Menu inicial do jogo

3.1 Representação do Estado do Jogo

Na representação do tabuleiro de jogo usam-se listas de listas que apenas incluem átomos para os diferentes tipos de peças (*red* e *blue*) e a casa vazia (*empty*). Para simplificação do desenvolvimento do jogo, escolheu-se a versão mais pequena do tabuleiro 8x8 com um total de 64 damas no início do jogo. O tabuleiro por sua vez é um elemento de uma lista que contém, além do tabuleiro, informação relativa ao estado do jogo, como o número de peças vermelhas e azuis, o modo de jogo, a situação de obrigação de salto e as coordenadas de uma posição da qual um salto é obrigatório.

3.1.1 Representação do estado inicial do tabuleiro:

```
[[blue,red,blue,red,blue,red,blue,red],
 [red,blue,red,blue,red,blue,red,blue],
 [blue,red,blue,red,blue,red,blue,red],
 [red,blue,red,blue,red,blue,red,blue],
 [blue,red,blue,red,blue,red,blue,red],
 [red,blue,red,blue,red,blue,red,blue],
 [blue,red,blue,red,blue,red,blue,red],
 [red,blue,red,blue,red,blue,red,blue]
]).
```

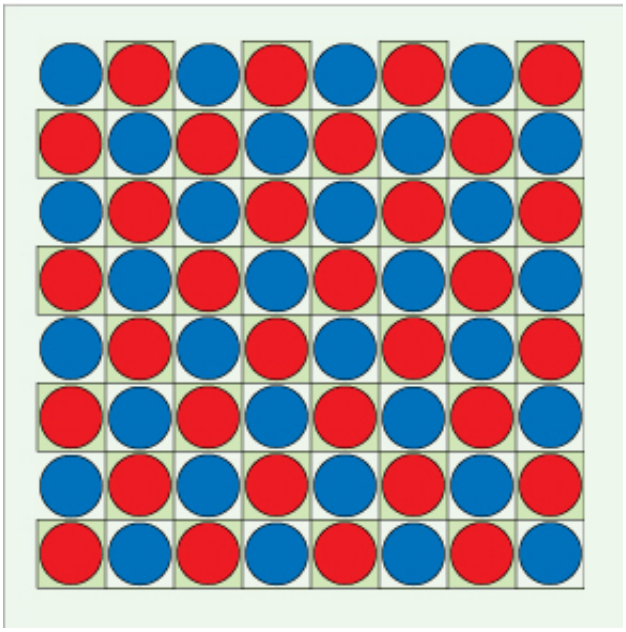

Figura 2: Estado inicial do jogo

3.2 Visualização do Tabuleiro

O tabuleiro pode ser visualizado pela linha de comandos a cada nova jogada efetuada quer pelo utilizador quer pelo computador. É disponibilizada informação acerca das peças ainda em jogo, e auxiliares na seleção de jogadas como a letra representativa de uma coluna e um número para uma linha.

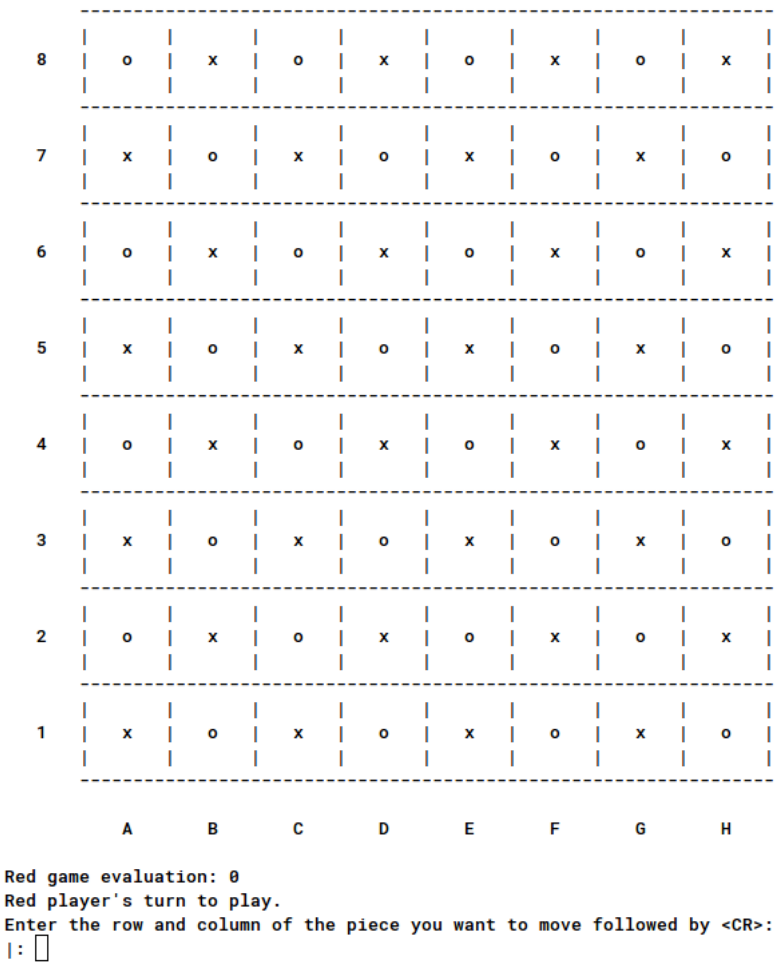

Figura 3: Estado inicial do jogo com pedido de seleção de jogada

3.3 Lista de Jogadas Válidas

Não está implementado nenhum método que angarie um conjunto de jogadas válidas, no entanto é possível determinar se existem movimentos válidos para uma dada peça do tabuleiro, sem movimentar nenhuma peça do tabuleiro, através de uma chamada à função `check_move_availability`.

```
check_move_availability(SrcRow, SrcCol, Player, Board):-
  % a move must be checked in all directions
  IncRow is SrcRow + 1,
  DecRow is SrcRow - 1,
  IncCol is SrcCol + 1,
  DecCol is SrcCol - 1,
  (
     IncRow =< 7, validate_move(SrcRow, SrcCol, IncRow, SrcCol,
         Player, Board, _, _);
     DecRow >= 0, validate_move(SrcRow, SrcCol, DecRow, SrcCol,
         Player, Board, _, _);
     IncCol =< 7, validate_move(SrcRow, SrcCol, SrcRow, IncCol,
         Player, Board, _, _);
     DecCol >= 0, validate_move(SrcRow, SrcCol, SrcRow, DecCol,
         Player, Board, _, _);
```

```
    DecRow >= 0, DecCol >= 0, validate_move(SrcRow, SrcCol,
        DecRow, DecCol, Player, Board, _, _);
    IncRow =< 7, IncCol =< 7, validate_move(SrcRow, SrcCol,
        IncRow, IncCol, Player, Board, _, _);
    DecRow >= 0, IncCol =< 7, validate_move(SrcRow, SrcCol,
        DecRow, IncCol, Player, Board, _, _);
    IncRow =< 7, DecCol >= 0, validate_move(SrcRow, SrcCol,
        IncRow, DecCol, Player, Board, _, _)
), !.
```

3.4 Execução de Jogadas

No caso da seleção de uma jogada manualmente através da consola, com a introdução de uma letra a representar a coluna e um número a representar uma linha, o programa tenta validar e mover uma peça do tabuleiro, tentando primeiro fazer uma jogada do tipo de salto. Quando o salto falha, são tentados outros dois tipos de movimentos: um adjacente e por fim, em último recurso, um movimento centralizador. Caso nenhum movimento seja válido, de acordo com as regras, assume-se que o jogador tem de dar a vez ao seu adversário. Em nenhum caso o jogo pode ficar numa situação na qual nenhum dos dois jogadores está sem jogadas válidas para executar.

```
make_move(SrcRow,SrcCol, DestRow, DestCol, Game, ModifiedGame):-
  (
    nl, write('Attempting to make a jump move...'), nl,
    make_jump(SrcRow,SrcCol, DestRow, DestCol, Game,
        TemporaryGame);

    write('Failed to make a jump move!'), nl, nl,
    write('Attempting to make an adjoining move...'), nl,
    make_adjoining_move(SrcRow,SrcCol, DestRow, DestCol, Game,
        TemporaryGame);

    write('Failed to make an adjoining move!'), nl, nl,
    write('Attempting to make a centering move...'), nl,
    make_centering_move(SrcRow,SrcCol, DestRow, DestCol, Game,
        TemporaryGame);

    write('Failed to make a centering move!'), nl, nl,
    get_board(Game, Board), get_player_turn(Game, Player),
    check_move_availability(SrcRow, SrcCol, Player, Board),
        ModifiedGame = Game;

    write('No valid moves were available -> Switching player
        turn!'), nl, nl,
    change_player_turn(Game,ModifiedGame), true
  ),
get_force_jump(TemporaryGame, ForceJumpMode),
(
   ForceJumpMode == noForceJump -> change_player_turn(
       TemporaryGame,ModifiedGame),! ;
   ModifiedGame = TemporaryGame
).
```

3.5 Avaliação do Tabuleiro

Apesar de ser feita uma avaliação simples do estado do jogo, não é feito nenhum aproveitamento para além da visualização deste valor no início de cada jogada. É feito um cálculo da avaliação do tabuleiro para um dado jogador, com a diferemça do seu número de peças com o número de peças do adversário. Neste

jogo em específico, é um cálculo relativamente acertado, ignorando os casos em que, no turno em vigor é possível fazer múltiplos saltos, sendo a avaliação um valor subestimado.

```
get_evaluation(Game,Player,Evaluation):-
    get_num_red_pieces(Game, NumRedPieces),
    get_num_blue_pieces(Game, NumBluePieces),
    (
       Player == redPlayer -> Evaluation is NumRedPieces -
           NumBluePieces;
       Player == bluePlayer -> Evaluation is NumBluePieces -
           NumRedPieces
    ),!.
```

3.6 Final do Jogo

Em cada iteração do ciclo principal do jogo, é feita a verificação do número de peças no tabuleiro. Quando um ou mais jogadores tiver zero peças em cima do tabuleiro o jogo está terminado e o vencedor foi o último jogador que fez uma movimentação no tabuleiro. É de salientar a possibilidade de o tabuleiro final estar completamente vazio, sendo o vencedor aquele que fez o salto final e que eliminou uma peça inimiga para fora do tabuleiro.

```
validate_board_pieces(Game):-
    get_num_red_pieces(Game,NumRedPieces),
    get_num_blue_pieces(Game,NumBluePieces),
    NumRedPieces > 0,
    NumBluePieces > 0,!.
```

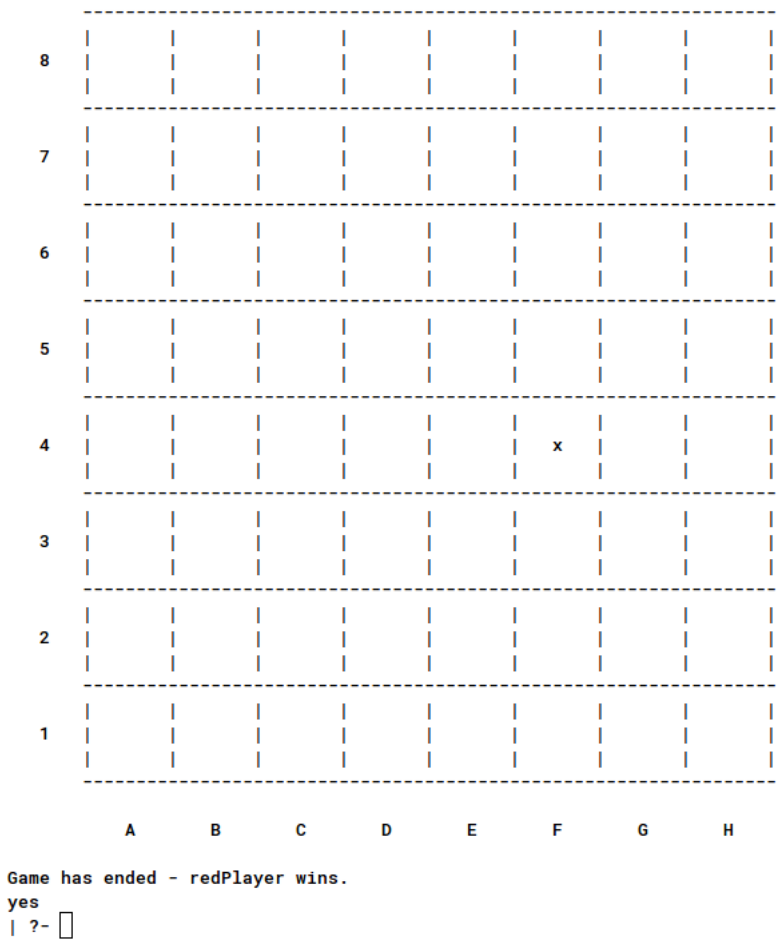

Figura 4: Estado final de um jogo com declaração de vencedor

3.7 Jogada do Computador

Não está implementada a possibilidade de seleção do modo de dificuldade de uma jogada do computador. O computador executa jogadas de forma aleatória.

4 Interface com o Utilizador

Descrever o módulo de interface com o utilizador em modo de texto.

5 Conclusões

Após a realização deste projeto, concluí-mos que ainda temos muito pouco à vontade no desenvolvimento de procedimentos de programação em lógica e que muitos dos hábitos herdados de programação de outras linguagens nos trouxeram muitas situações ante problemas dos quais ainda não sabemos como contornar. É também de criticar a complexidade exagerada do trabalho, considerando o nosso conhecimento limitado e inexperiência na linguagem em questão, sendo

que propunhamos este nível de complexidade apenas a partir de um segundo trabalho, ou um prazo de entrega mais alargado para este primeiro projeto.

Referências

[1] Sterling, Leon *The Art of Prolog*, The MIT Press 2nd edition, 2000.

[2] Abstract games, `http://www.marksteeregames.com/MSG_abstract_games.html`, 14 10 2016.

[3] Cage rules, `http://www.marksteeregames.com/Cage_rules.html`, 14 10 2016.

A Código fonte

Código Prolog implementado devidamente comentado e outros elementos úteis que não sejam essenciais ao relatório.