

**Azacru\_1**

**Relatório Final**

Mestrado Integrado em Engenharia Informática e Computação

**Programação em Lógica**

**Grupo azacru\_1:**

Nuno Miguel Outeiro Pereira – up201506265

Ricardo José Santos Pereira- up201503716

Faculdade de Engenharia da Universidade do Porto

Rua Roberto Frias, sn, 4200-465 Porto, Portugal

12 de novembro de 2017

**Resumo**

O trabalho desenvolvido consiste num jogo de tabuleiro para ser jogado entre duas pessoas denominado azacru, um dos jogos pertencentes ao grupo pacru, que tem como espaço de execução a linha de comandos.

Trata-se de um jogo onde o objetivo é deixar o adversário incapaz de realizar qualquer tipo de jogada, como tal, é um jogo que requer alguma preparação e previsão mental.

Tendo sido, este jogo, desenvolvido em PROLOG, uma linguagem de índole bastante diferente da nossa zona de conforto, deparámo-nos com alguns problemas de implementação, mas que através de alguma pesquisa e consulta foram ultrapassados.

Após a implementação deste jogo, podemos, de facto, assumir que a nossa destreza e conhecimento em PROLOG aumentou, visto que nos deparamos com várias situações que obrigaram a busca de conhecimento e tentativas de implementação que contornassem os entraves que foram aparecendo.

**Conteúdo**

1. **Introdução 4**
2. **Azacru 5**
3. **Lógica do Jogo 8**
   1. Representação do Estado do Jogo . . . . . . . . . . . . . . . . . . 4
   2. Visualização do Tabuleiro . . . . . . . . . . . . . . . . . . . . . . 4
   3. Lista de Jogadas Válidas . . . . . . . . . . . . . . . . . . . . . . . 4
   4. Execução de Jogadas . . . . . . . . . . . . . . . . . . . . . . . . . 4
   5. Avaliação do Tabuleiro . . . . . . . . . . . . . . . . . . . . . . . . 4
   6. Final do Jogo . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
   7. Jogada do Computador . . . . . . . . . . . . . . . . . . . . . . . 4
4. **Interface com o Utilizador 12**
5. **Conclusões 12**

**Bibliografia 12**

**A Nome do Anexo 13**

# 1 Introdução

No âmbito da unidade curricular de Programação em Lógica, foi-nos concebida a tarefa de implementar um de uma lista de jogos de tabuleiro. O jogo selecionado por nós foi o Azacru.

O que nos motivou à sua implementação foi o aumento de conhecimento que ele proporcionou pois não conhecíamos o jogo nem nenhum do pacote de jogos Pacru. É um jogo que, como qualquer jogo de tabuleiro, obriga a prática e previsão mental, assim como à implementação de uma estratégia.

Os principais objetivos deste trabalho baseiam-se na consolidação de conhecimentos adquiridos durante o semestre.

Este relatório encontra-se estruturado em várias secções como é possível detetar pelo índice. Existem divisões para a história e regras do jogo, para a lógica por detrás do jogo assim como para a sua implementação em PROLOG, para o modo de interação com o utilizador, para conclusões retiradas após elaboração do trabalho e também secções para a bibliografia e anexos do código desenvolvido.

# 2 Azacru

# História

O Azacru é um jogo que foi lançado em 2005 como parte integrante de um conjunto de jogos chamado Pacru Series 302 que continha vários jogos que partilhavam o mesmo tabuleiro, assim como as mesmas peças e marcos, conjunto este inventado por Mike Wellman.

Este jogo tem também o seu mestre, que neste caso é Martyn J.Hamer que já foi múltiplas vezes campeão do mundo.

# Regras do Jogo

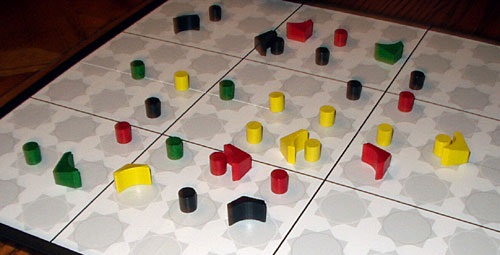
O Azacru é um jogo multi-jogador no qual cada um dos jogadores tem peças e marcadores de cor diferente. A ordem pela qual começam a jogar é deixada ao critério dos jogadores. Neste trabalho, vamos apenas implementar o jogo para 2 jogadores, sendo que por isso, cada jogador começa com 4 peças.

O jogo vai-se jogando alternadamente entre os jogadores, sendo que só é possível passar a vez caso o jogador esteja numa posição onde não consegue realizar qualquer jogada.

Os movimentos permitidos à peça são 3, sendo eles, em relação à posição da peça original, em frente, 45° à esquerda ou 45° à direita. Ao fim da jogada a peça fica a apontar na última direção em que foi jogada, sendo também colocado um marcador nessa posição do tabuleiro, exceto quando no final da jogada a peça tiver parado noutro sector.

O poder de alcance da peça numa jogada depende do sector em que se localiza, sendo que é o mesmo em casas à soma do número de marcadores da sua cor nesse sector. Quando não existem marcadores da cor da peça nesse sector o poder de alcance da peça é 1. Não é obrigatório percorrer todas as casas até ao poder máximo, a jogada pode ser de 1 casa até o número do alcance máximo.

Em baixo segue uma foto dum tabuleiro aquando da realização de um jogo entre 4 jogadores para se perceber melhor do que se fala quando se referem marcadores e sectores.

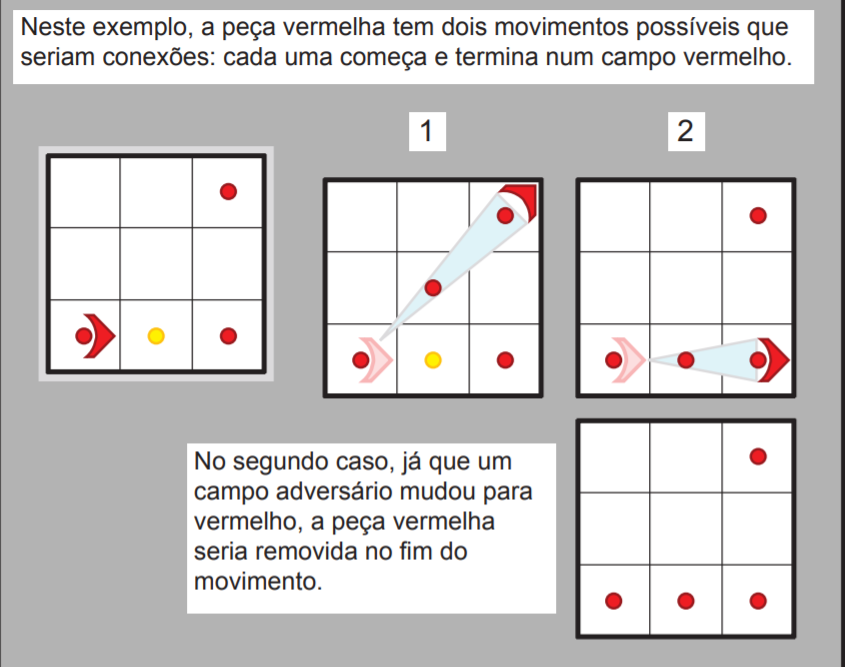


Quando a jogada inclui atravessar mais do que uma casa, não é permitido trocar a direção a meio do movimento, a direção pela qual se inicia o movimento é a direção pela qual a peça passará todas as casas.

Se no final da jogada a peça tiver passado para outro sector pode não terminar o seu movimento na posição em que iniciou a jogada, sendo que neste caso o jogador pode rodar a peça 45° no final da jogada quer seja para a esquerda ou para a direita.

Quando o jogador realiza um movimento da sua peça entre um campo da sua cor e outro dessa mesma cor, e não se passou através de uma peça, faz com que todos os campos intermédios fiquem também da cor da sua peça. Esta jogada pode também ser realizada saltando peças, contudo as casas intermédias entre a 1ª e a última não iram se alterar.

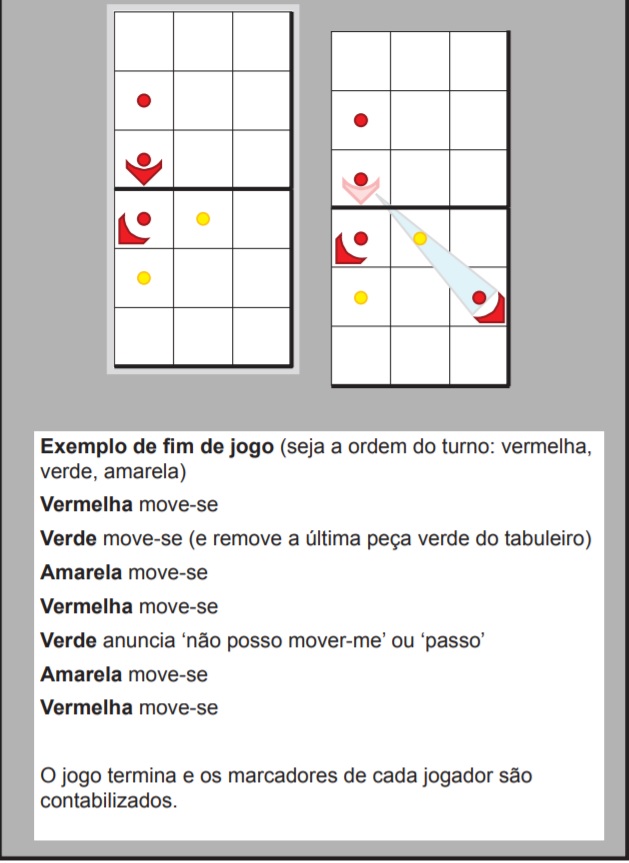
Quando, na jogada, são trocadas casas de cor do adversário para a cor da peça dessa jogada, a peça é então extraída do tabuleiro.



É proibido mover uma peça para uma casa com a cor do adversário, não se pode também mover uma peça para um campo ocupado por outra. Só é permitido saltar por cima de uma peça, se a casa onde se for parar já for da cor da peça jogada, de outra forma não é possível.

O jogo termina quando um dos jogadores não for capaz de realizar nenhuma jogada ou quando todas as suas peças foram retiradas do jogo. Quando tal se sucede, o jogador passa a vez, os restantes jogadores jogam mais um turno e depois disso contabilizam-se os marcadores de cada jogador e quem tiver mais acumulados ganha o jogo.

De seguida, segue-se um exemplo de um término de jogo:



# 3 Lógica do Jogo

Descrever o projeto e implementa¸c˜ao da l´ogica do jogo em Prolog, incluindo a forma de representac¸˜ao do estado do tabuleiro e sua visualiza¸c˜ao, execuc¸˜ao de movimentos, verificac¸˜ao do cumprimento das regras do jogo, determinac¸˜ao do final do jogo e c´alculo das jogadas a realizar pelo computador utilizando diversos n´ıveis de jogo. Sugere-se a estrutura¸c˜ao desta sec¸c˜ao da seguinte forma:

## 3.1 Representação do Estado do Jogo

Detalhando agora a implementação em Prolog do jogo, o tabuleiro definir-se-á como uma lista de listas, especificamente nove listas de nove átomos, onde os átomos serão números inteiros representativos de cada campo no tabuleiro, e contendo informação sobre qualquer peça ou marcador que se possa lá encontrar. Os números serão constituídos por um ou três dígitos e terão a forma de:

Átomo = OPM

* O é um número entre 1 e 8, correspondendo à orientação da peça.
* P é um número entre 1 e 2, correspondendo à identidade da peça que aí se encontra (caso pertença ao jogador 1 ou 2).
* M é um número entre 0 e 2, correspondendo à identidade do marcador que aí se encontra (sendo 0 caso o campo esteja vazio).
* Se não estiver nenhuma peça presente nesse campo, então O e P serão substituídos por 0 (daí um número com apenas um dígito).

Deste modo, os estados iniciais, intermédios e finais do tabuleiro podem ser descritos por algo como as seguintes listas, com o respetivo output:

Inicial

[[420,000,510,000,000,000,520,000,000],

[000,000,000,000,000,000,000,000,000],

[000,000,000,000,000,000,000,000,000],

[000,000,000,000,000,000,000,000,000],

[310,000,000,000,000,000,000,000,720],

[000,000,000,000,000,000,000,000,000],

[000,000,000,000,000,000,000,000,000],

[000,000,000,000,000,000,000,000,000],

[000,000,110,000,000,000,120,000,810]]

Intermédio

[[000,000,000,000,211,000,520,000,000],

[000,422,000,001,000,000,000,000,000],

[000,000,000,000,000,000,000,000,000],

[000,211,000,000,000,822,000,000,000],

[000,000,000,000,000,000,002,002,000],

[000,000,000,000,000,000,000,000,000],

[000,111,000,000,000,000,000,000,000],

[000,001,000,000,000,000,122,000,000],

[000,000,000,000,000,000,000,000,810]]

Final

[[000,001,001,001,722,002,002,002,000],

[001,001,001,002,001,001,001,001,002],

[001,002,002,001,000,001,001,001,002],

[001,002,111,002,000,001,001,001,002],

[001,001,001,002,002,001,001,002,002],

[001,002,001,001,001,000,001,001,002],

[001,001,002,002,002,001,001,000,002],

[001,001,000,001,322,002,002,002,001],

[001,000,001,002,002,002,211,001,001]]

## 3.2 Visualização do Tabuleiro

O tabuleiro é representado através de chamadas a *write*/1 com átomos que consistem em strings de caracteres ASCII. Estas chamadas são sequenciadas através de uma série de predicados que primeiro efetuam o processamento do input e só depois procedem ao *display* dos átomos resultantes. O predicado que faz a chamada apropriada destes predicados é *print\_matrix*/1, que deve receber a lista referida no capítulo anterior (esta lista deve ter exatamente nove listas de nove átomos).

Devido à natureza do jogo, quer no *display* quer nos predicados é transversal a complexidade, já que no primeiro caso recorre-se apenas a números para se efetuar a representação das peças do tabuleiro. A título de exemplo, a chamada ao predicado para visualizar o tabuleiro sem qualquer peça (vazio) teria como argumento:

print\_matrix([[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], [0,0,0,0,0,0,0,0,0],[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],[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0]]).

Produzindo o seguinte output:

Como se pode perceber pela imagem, cada campo é constituído por um espaço de 7 por 3 carateres, sendo no entanto apenas utilizado um espaço de 3 por 3 caracteres. Assim, conseguimos exprimir elementos como este:

Que representa um campo com um marcador e uma peça do jogador 2 com orientação sudeste.

**FALTA INTRODUZIR IMAGENS**

## 3.3 Lista de Jogadas Válidas

Obten¸c˜ao de uma lista de jogadas poss´ıveis. Exemplo: *valid moves(+Board, -ListOfMoves)*.

## 3.4 Execução de Jogadas

Validac¸˜ao e execu¸c˜ao de uma jogada num tabuleiro, obtendo o novo estado do jogo. Exemplo: *move(+Move, +Board, -NewBoard)*.

### 3.5 Avaliação do Tabuleiro

Avalia¸c˜ao do estado do jogo, que permitir´a comparar a aplica¸c˜ao das diversas jogadas dispon´ıveis. Exemplo: *value(+Board, +Player, -Value)*.

## 3.6 Final do Jogo

Verificac¸˜ao do fim do jogo, com identifica¸c˜ao do vencedor. Exemplo: *game over(+Board, -Winner)*.

## 3.7 Jogada do Computador

Escolha da jogada a efetuar pelo computador, dependendo do n´ıvel de dificuldade. Por exemplo: *choose move(+Level, +Board, -Move)*.

# 4 Interface com o Utilizador

É possível iniciar o jogo com o comando start\_game, sendo que será apresentado o tabuleiro inicial, seguindo-se o menu para o jogador escolher peça que pretende mover. Após essa escolha, é pedido ao utilizador para indicar se se pretende mover para a frente, indicando a letra “f” seguida do número de casas a avançar, sendo que este número vai ser condicionado pelo seu poder de movimento naquele momento do jogo, ou “r” ou “l” consoante queria fazer o movimento de 45º para a direita ou esquerda, respetivamente, sendo também acrescentado à frente o número de casas.

O programa responde se é uma jogada válida e, de facto, a efetua ou, caso contrário, é apresentada uma mensagem a indicar que é uma jogada inválida, pedindo novo input ao utilizador.

É dada a vez ao outro jogador, e a partir daqui as peças passam a estar atribuídas a cada um dos jogadores, sendo que se um jogador tentar mover as peças do outro, o programa apresenta a mensagem de erro “Invalid piece”, pedindo de novo input para escolher uma peça válida.

O programa vai alternando entre jogadores, seguindo-se o esquema aqui apresentado.

**5 Conclusões**

Este projeto foi bastante útil visto que forçou a adquirição de novos conhecimentos e aumento da destreza em PROLOG.

Apesar da escassez de tempo com que fomos enfrentados, podemos assumir que concluímos o trabalho conforme planeado e que o balanço é positivo.

# Bibliografia

# Sites:

* <http://www.boardability.com/game.php?id=azacru>
* <http://www.pacru.com/rulesPT.pdf>

# A Código desenvolvido

C´odigo Prolog implementado devidamente comentado e outros elementos u´teis que n˜ao sejam essenciais ao relat´orio.