U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Latrunculi XXI

Relatório

Mestrado Integrado em

Engenharia Informática e Computação

Programação em Lógica

Grupo Latrunculi_XXI_1:

Daniel Filipe Santos Marques – 201503822

João Filipe Lopes de Carvalho – 201504875

Porto, 11 de novembro de 2017

Resumo

A unidade curricular de Programação em Lógica desafiou o grupo a desenvolver um jogo de tabuleiro em Prolog. O tema escolhido foi o Latrunculi XXI, uma variante do Latrunculi clássico que, por sua vez, é da família do xadrez.

O trabalho teve como objetivo aplicarmos os conhecimentos aprendidos nas aulas da unidade curricular. A adaptação ao novo paradigma constituiu a maior dificuldade do trabalho. No entanto, com a consulta dos recursos fornecidos pelos docentes e da documentação do SICStus Prolog, este problema foi sendo ultrapassado, construindo novos predicados para a lógica do jogo e utilizando alguns pré-definidos pela implementação do SICStus.

O projeto foi finalizado e, como resultado, o grupo desenvolveu um jogo de simples aparência, mas que encobre uma lógica complexa e desafiante.

Concluímos que o paradigma a que fomos introduzidos tem um grande potencial e que, com o devido conhecimento, pode ser bastante poderoso e eficiente. Com este projeto, o nosso conhecimento relativo a este paradigma e, mais especificamente, à linguagem de Prolog aumentou substancialmente.

Índice

1. Introdução

No âmbito da unidade curricular de Programação em Lógica foi feito um projeto em Prolog com o objetivo de aplicar os conhecimentos do paradigma de programação em lógica transmitidos nas aulas. Foi dada a hipótese de escolha de uma lista pré-definida de temas, na qual selecionamos o Latrunculi XXI. Esta decisão baseou-se na interessante dinâmica do jogo, das múltiplas formas de capturar peças e do nível de estratégia necessário para se ser um bom jogador.

Este relatório tem o objetivo de transmitir todas as informações necessárias para disfrutar do jogo desenvolvido, bem como revelar a estrutura da nossa implementação. O relatório tem a seguinte estrutura:

- Descrição do Jogo – descreve a sua história e regras, exemplificadas com imagens.
- Implementação – descreve a forma como foi implementado em Prolog.
- Interface com o Utilizador – descreve o modo de visualização e a interação do programa com o utilizador, exemplificando com imagens.

2. Descrição do jogo

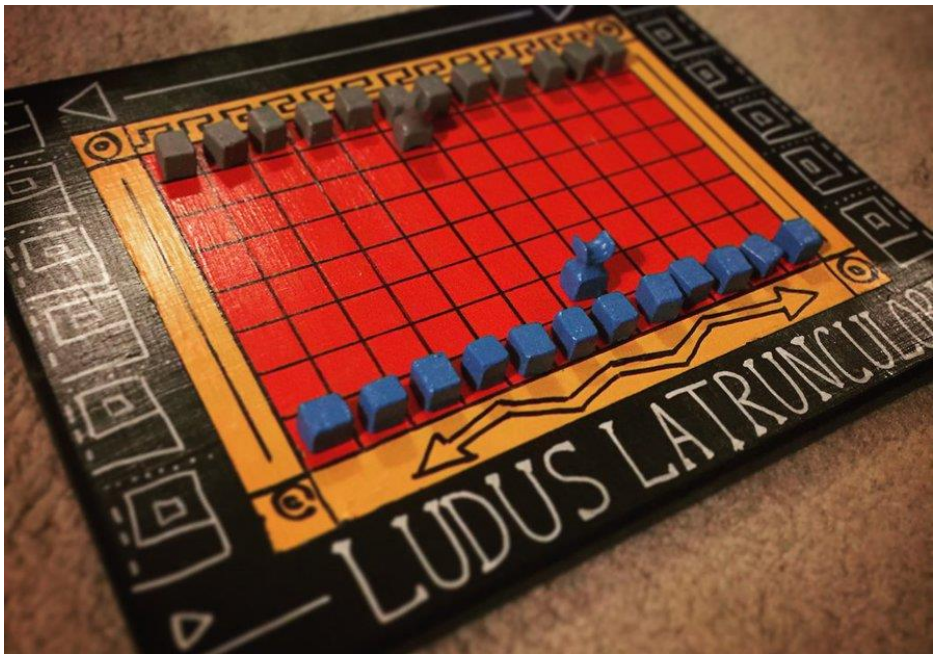
Latrunculli XXI é um jogo de estratégia da família do xadrez. Cada jogador tem na sua posse vários "Soldados" e uma peça da mesma cor, mas com uma forma diferente, que é o "Dux", também conhecido por Rei ou Líder.

Todas as peças se deslocam ortogonalmente e o objetivo do jogo é capturar o dux inimigo.

a. História

O Latrunculli Clássico (Ludus latrunculorum ou "Jogo dos Pequenos Soldados"), surgiu no antigo Império Romano, 27 AC – 1453 DC.

Foi então em 2015 que surgiu o Latrunculli XXI, quando um pai queria ensinar ao seu filho algumas variantes de xadrez e redesenhou um pouco o Latrunculli Clássico, de modo a ser mais fácil de iniciantes com base de xadrez entenderem o conceito do jogo e se tornar mais fácil de haver um vencedor, criando-se assim novas regras, nomenclaturas, táticas e estratégias.



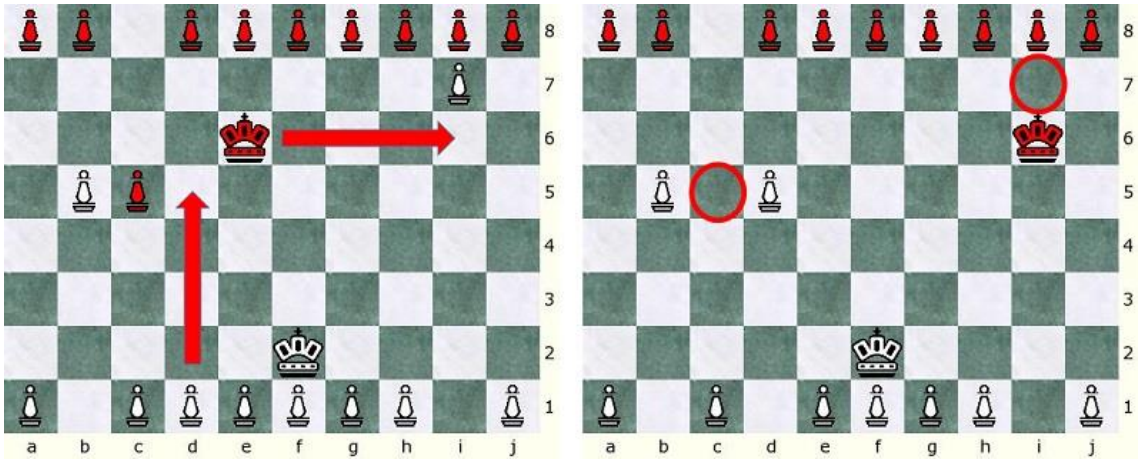
1 - Tabuleiro real de Latrunculi XXI

b. Regras

O jogo é constituído por um tabuleiro de dimensões 8x8 (1) e por 18 peças, 9 por jogador, sendo jogado por 2 jogadores. Existem 2 tipos de peças: duxes e soldados, existindo 1 e 8 peças, respetivamente, por jogador.

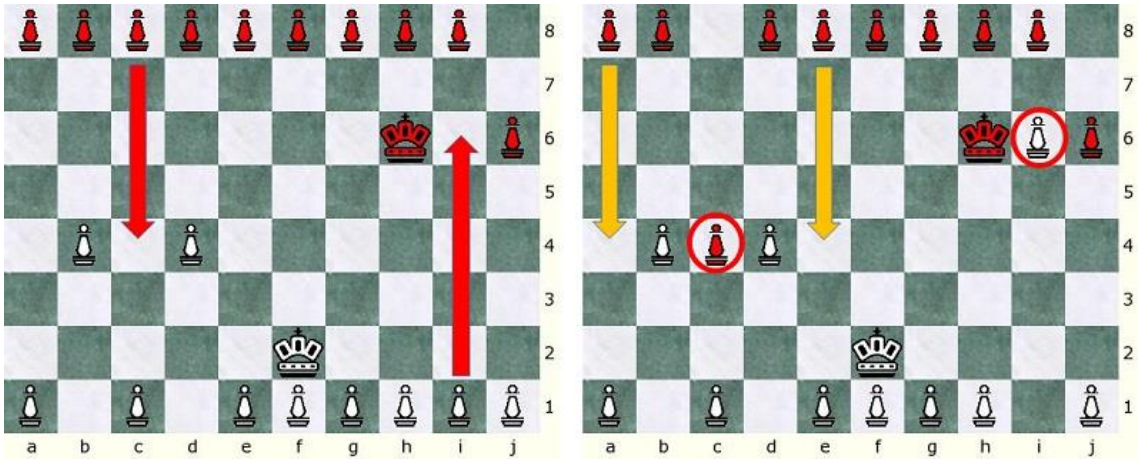
As peças movem-se ortogonalmente, ou seja, não são permitidas jogadas diagonais. Podem deslocar-se um número arbitrário de células, porém não podem saltar outras peças.

Para realizar a captura clássica de um soldado inimigo, é necessário que essa peça fique rodeada horizontalmente ou verticalmente por peças inimigas.



2 - Exemplo de Captura

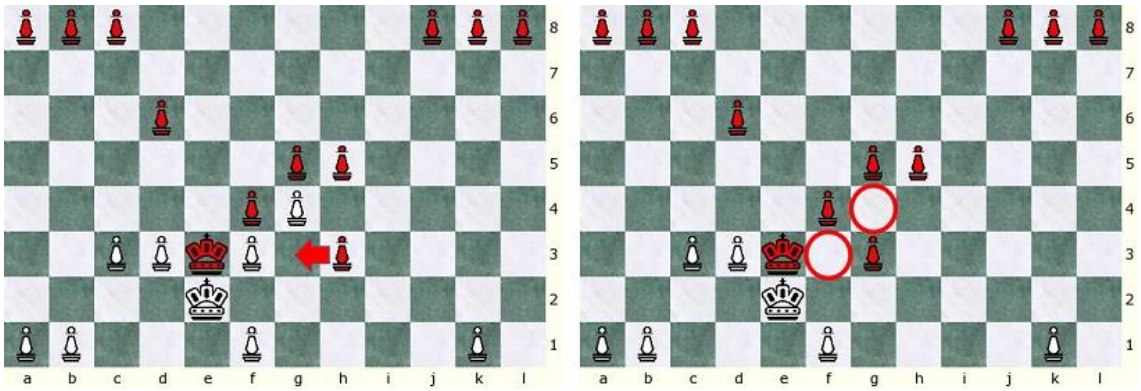
No entanto, é perfeitamente seguro uma peça mover-se para o meio de duas peças inimigas, não estando sujeitas a captura imediata (ou suicídio).



3 - Exemplo de soldados a deslocarem-se para o meio de duas peças inimigas sem risco imediato

(1) Várias escavações revelaram várias dimensões diferentes, mas a mais comum é a de 8x8.

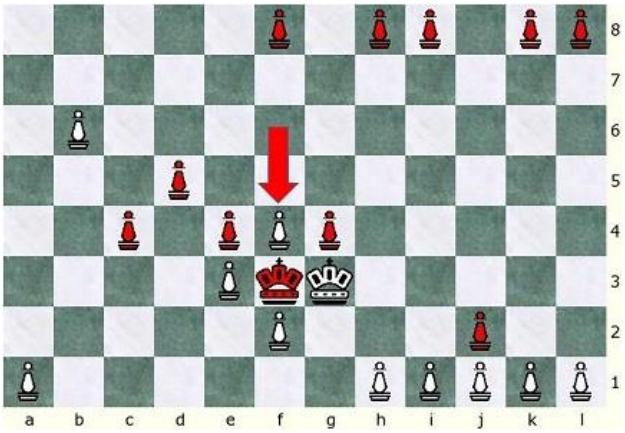
É também possível capturar múltiplas peças inimigas numa jogada.



4 - Exemplo de captura múltipla

Uma exceção à regra acontece quando a peça a capturar se encontra num dos cantos ou borda do tabuleiro. Neste caso, continuam a ser necessárias 2 peças a rodeá-la ortogonalmente.

O objetivo do jogo é imobilizar o dux do oponente ou capturar todos os soldados do inimigo. Um dux diz-se imobilizado se todas as células adjacentes a este se encontrarem peças bloqueando todas as direções. É proibido alguém imobilizar o seu próprio dux.



5 - Exemplo de captura de um dux

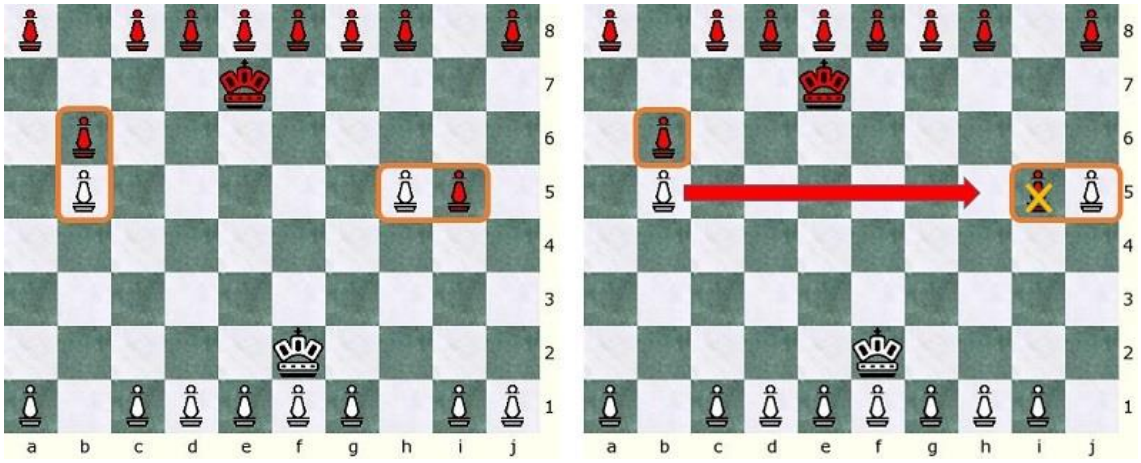
c. Regras Específicas

Em adição às regras tradicionais, a versão Latrunculi XXI, possui um conjunto adicional de regras e definições.

As jogadas estão em divididas em ofensivas e defensivas. As primeiras são constituídas por capturas e ataques ao dux inimigo (bloqueio de uma das direções ortogonais deste). As últimas são todas as outras jogadas, tais como reposicionamento.

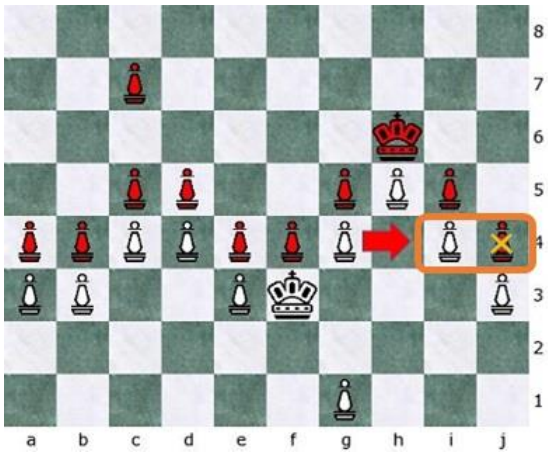
Chama-se conflito a uma situação em que uma peça está adjacente a uma inimiga. Um grupo de 3 ou mais peças envolvidas num conflito chama-se uma disputa.

Bloqueio de Soldados: Um soldado num conflito encontra-se bloqueado, isto é, quando uma peça inimiga se encontra numa posição ortogonal adjacente e, por esta razão, não pode efetuar jogadas defensivas. Podem, no entanto, desbloquear-se de modo a efetuar uma jogada ofensiva. Qualquer peça que seja parte de uma disputa pode ser reposicionada desde que a peça inimiga que era vítima da disputa continue bloqueada.

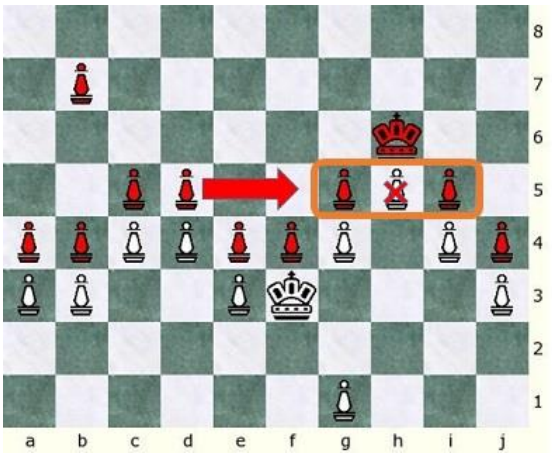


6 - Exemplo de um soldado bloqueado, mas movendo-se para fazer uma jogada ofensiva

Empurrão e esmagamento: Uma peça pode empurrar uma peça amiga de modo a esmagar uma inimiga contra as bordas do tabuleiro ou contra outra peça amiga, de modo a capturar um soldado inimigo (esta técnica não se aplica à captura de duxes). O movimento do empurrão deve ser na mesma direção formada pelas peças amiga(s) e inimiga.

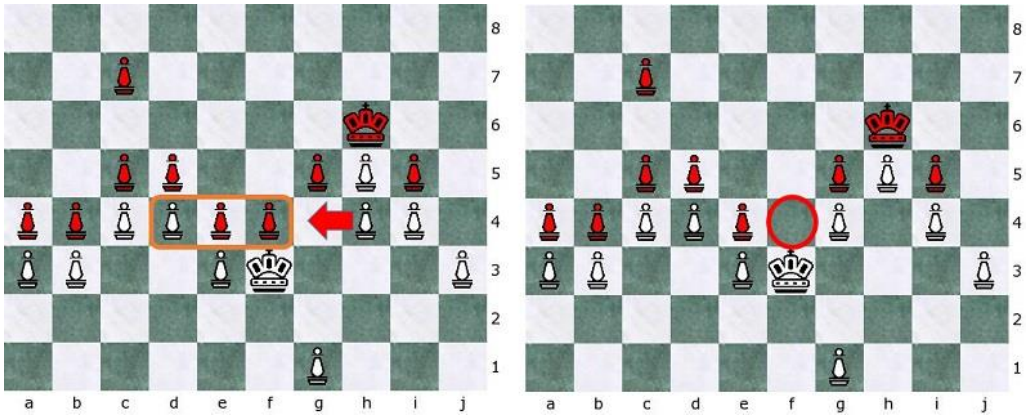


7 - Exemplo da técnica contra as bordas



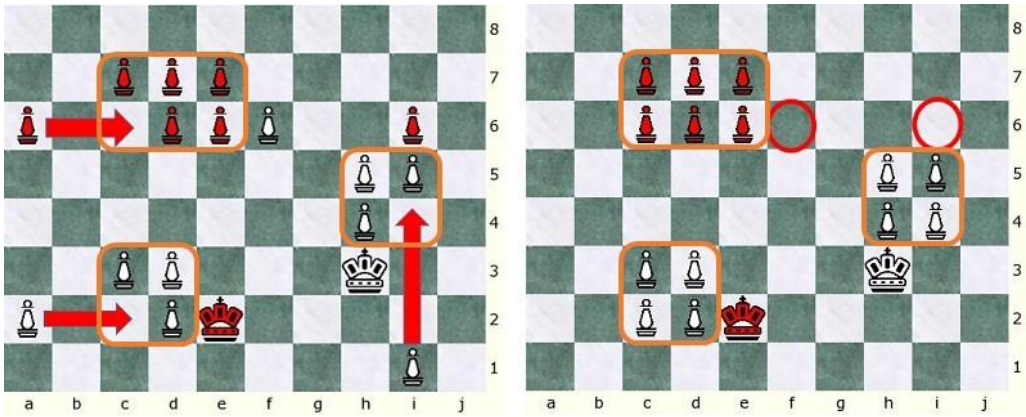
8 - Exemplo da técnica contra uma peça amiga

Ataque lateral: O soldado na ponta de uma defesa linear pode ser capturado por um ataque lateral – uma peça amiga move-se em direção a este (ao fim da formação linear), encurralando todas essas peças inimigas entre duas peças amigas e capturando o último.



9 - Exemplo de um ataque lateral

Ataque Phalanx: Este é um ataque do qual resulta um Testudo e a captura de um soldado na mesma direção, mas no outro lado do Testudo. Um Testudo é uma formação defensiva retangular de peças. Este ataque permite apenas a captura de um soldado, porém duxes podem participar nestes.



10 - Exemplos de ataques Phalanx

Mobilidade do dux: Considera-se que o dux tem direções bloqueadas apenas quando pelo menos uma é bloqueada por uma peça inimiga. Caso contrário, mesmo que bloqueado por peças amigas, o dux consegue mover-se livremente (nas direções desobstruídas). A mobilidade do dux pode ser resumida pela seguinte tabela:

Direções bloqueadas \ Posição do Dux	Canto do tabuleiro	Bordas do tabuleiro	Outras posições
0	Mobilidade Total	Mobilidade Total	Mobilidade Total
1	Bloqueado	Um quadrado ou jogada ofensiva	Um quadrado ou jogada ofensiva
2	Checkmate	Bloqueado	Jogada ofensiva
3		Checkmate	Bloqueado
4			Checkmate

11 - Tabela de mobilidade do dux

3. Implementação

A nossa implementação de Latrunculi XXI está dividida em 7 módulos: *board*, *bot*, *captures*, *display*, *game*, *moves* e *utilities*. Em adição, existe um módulo de testes unitários (*tests*) para confirmar o correto funcionamento de cada regra do jogo. Os módulos mais importantes são: *moves* – responsável por verificar a validade de uma jogada e efetuá-la; *captures* – responsável por verificar captura de peças numa jogada; *bot* – responsável pelas jogadas do computador considerando a dificuldade selecionada.

a. Representação do Estado do Jogo

O tabuleiro, as peças e os jogadores são definidos no ficheiro *board.pl* bem como os predicados que sobre nestes atuam. O tabuleiro é representado por uma lista de listas de 8 elementos, constituindo uma matriz de 8x8. Estes elementos são peças (átomos) que pertencem a um jogador. De todos os predicados do módulo salienta-se o *initialBoard(-Board)* que inicializa o tabuleiro com o seu estado inicial. As peças correspondem aos átomos *black_soldier*, *white_soldier*, *black_dux* e *white_dux*. Os jogadores são representados pelos valores numéricos *1* e *2*.

b. Visualização do Tabuleiro

Os predicados responsáveis pela visualização do tabuleiro, bem como a dos menus, na consola do interpretador de Prolog encontram-se no ficheiro *display.pl*. Para imprimir um tabuleiro executa-se o predicado *drawBoard(+Board)*. Para visualizar os menus basta chamar os predicados *printMenu/0* e *printDifficultyMenu/0*. Todos estes predicados fazem uso do predicado pré-implementado *write/1*.

c. Lista de Jogadas Válidas

O módulo de inteligência artificial – *bot.pl* – implementa o predicado *getAllMoves(+Board, +PlayerNumber, -PossibleMoveList)* que obtém todas as jogadas que o jogador *PlayerNumber* pode fazer num determinado tabuleiro *Board*, colocando-as na lista *PossibleMoveList*. Em cada jogada gerada é verificada a sua validade antes de ser adicionada à lista recorrendo ao predicado *move/7* e verificando se este teve sucesso.

d. Execução de Jogadas

A validação, execução e pós-processamento de uma jogada são efetuados no ficheiro *moves.pl* pela chamada ao predicado *move(+Board, +Xi, +Yi, +Xf, +Yf, -NewBoard).* Este predicado verifica se todas as regras do jogo estão a ser cumpridas através de chamadas a predicados definidos neste mesmo módulo: *isOrthogonal/4*, *isElementBetween/5*, *checkLockedSoldier/5*, *checkDuxMobility/5* e *friendDuxImmobilized/5*. A peça é movida, criando um novo tabuleiro baseado em *Board*. Finalmente, são verificadas capturas de peças segundo as regras clássicas e também as do século XXI, utilizando os predicados *captureClassic/4* e *captureXXI/6*, respetivamente.

e. Avaliação do Tabuleiro

A avaliação de uma jogada é feita em relação ao efeito que esta terá no tabuleiro e às propriedades da jogada, ao invés do estado atual do tabuleiro. Esta avaliação acontece recorrendo aos predicados *gameIsOver(+Board)* e *moveIsOffensive(+Board, +Xi, +Yi, +Xf, +Yf)*. O primeiro é verificado depois de uma jogada acontecer, enquanto que a última pode ser verificada sem que a jogada seja feita. Estes predicados são utilizados com intenção de avaliação no módulo de inteligência artificial, no caso de maior dificuldade. Neste caso atribui-se um valor mais elevado a uma jogada que termine o jogo em relação a uma jogada ofensiva que, por sua vez, tem maior valor que as restantes jogadas.

f. Final do Jogo

O final de jogo é verificado através do predicado *gameIsOver(+Board)* ou *gameIsOver(+Board, -Winner)*, definidos no ficheiro *capture.pl*. Este predicado sucede quando o dux de um jogador é capturado ou todos os seus soldados foram capturados. Visto que as peças são capturadas pelo predicado *move/6*, as condições de fim de jogo podem ser feitas recorrendo a um predicado utilitário *findMatrixElement(+Board, +Piece)*.

g. Jogada do Computador

No projeto foram implementados dois tipos de dificuldade: ingénuo (*dumb*) e inteligente (*intelligent*). Em ambos os casos, numa jogada do computador são geradas todas as jogadas possíveis – predicado *getAllMoves/3*. Em seguida, é escolhida uma jogada. Este passo é o único que depende da dificuldade e é efetuado pelo predicado *pickMove(+Difficulty, +Board, +MoveList, -Move)*. No caso do *dumb bot*, a escolha é aleatória fazendo uso do predicado pré-definido *random/3*, do qual se retira o índice da jogada escolhida. No caso do *intelligent bot*, a escolha é hierárquica, atribuindo respetivamente valores decrescentes de prioridade aos seguintes tipos de jogadas: jogadas que terminam o jogo – selecionadas através dos predicados *findGameOverMove/3* e *gameIsOver/1*; jogadas ofensivas – selecionadas através dos predicados *findOffensiveMove/3* e *moveIsOffensive/5*; outras jogadas – a sua seleção é aleatória.

4. Interface com o Utilizador

A interface com o utilizador foi mantida simples, permitindo jogar mesmo quando o utilizador se engana (coordenadas ou jogadas inválidas) e mostrando informações consideradas relevantes quanto às jogadas do computador. Em *inputs* é esperado um número seguido de um ponto, submetido ao programa premindo a tecla *Enter*. Em outros casos, devidamente sinalizados, é pedido ao utilizador que apenas prima a tecla *Enter* para continuar.

```
---------------------------------
-        Latrunculi XXI         -
---------------------------------
-                               -
-   1. Player vs Player         -
-   2. Player vs Computer       -
-   3. Computer vs Computer     -
-   4. Exit                     -
-                               -
---------------------------------
Choose a game mode:
|:
```

12 - Menu Principal

```
    0   1   2   3   4   5   6   7
  _____________________________________
0 |   | x | x | x | x | x | x | x |
  |___|___|___|___|___|___|___|___|
1 |   |   |   | X |   |   |   |   |
  |___|___|___|___|___|___|___|___|
2 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
3 | x |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
4 | o |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
5 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
6 |   |   |   |   | 0 |   |   |   |
  |___|___|___|___|___|___|___|___|
7 |   | o | o | o | o | o | o | o |
  |___|___|___|___|___|___|___|___|

Player 1
  Enter source coordinates:
X?|: 1.
Y?|: 7.
  Enter destination coordinates:
X?|: 1.
Y?|: 3.
```

```
    0   1   2   3   4   5   6   7
  _____________________________________
0 |   | x | x | x | x | x | x | x |
  |___|___|___|___|___|___|___|___|
1 |   |   |   | X |   |   |   |   |
  |___|___|___|___|___|___|___|___|
2 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
3 |   | o |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
4 | o |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
5 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
6 |   |   |   |   | 0 |   |   |   |
  |___|___|___|___|___|___|___|___|
7 |   |   | o | o | o | o | o | o |
  |___|___|___|___|___|___|___|___|

Player 2
  Enter source coordinates:
X?|: 1.
Y?|: 0.
  Enter destination coordinates:
X?|: 1.
Y?|: 2.
```

13 - Exemplo de uma sequência de jogadas de jogadores

```
---------------------------------
-        Latrunculi XXI         -
---------------------------------
-                               -
-   1. Dumb Bot                 -
-   2. Intelligent Bot          -
-                               -
---------------------------------
Choose a bot difficulty:
|:
```

14 - Menu de Dificuldade

```
    0   1   2   3   4   5   6   7
  _________________________________
0 | x | x | x | x | x | x | x | x |
  |___|___|___|___|___|___|___|___|
1 |   |   |   | X |   |   |   |   |
  |___|___|___|___|___|___|___|___|
2 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
3 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
4 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
5 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
6 |   |   |   |   | O |   |   |   |
  |___|___|___|___|___|___|___|___|
7 | o | o | o | o | o | o | o | o |
  |___|___|___|___|___|___|___|___|

Computer 1
Press [Enter] to continue.
|:
  Generating moves...
  Generated 53 moves.
  Moving from (4,6) to (4,1).
Press [Enter] to continue.
|:
```

```
    0   1   2   3   4   5   6   7
  _________________________________
0 | x | x | x | x | x | x | x | x |
  |___|___|___|___|___|___|___|___|
1 |   |   |   | X | O |   |   |   |
  |___|___|___|___|___|___|___|___|
2 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
3 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
4 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
5 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
6 |   |   |   |   |   |   |   |   |
  |___|___|___|___|___|___|___|___|
7 | o | o | o | o | o | o | o | o |
  |___|___|___|___|___|___|___|___|

Computer 2
Press [Enter] to continue.
|:
  Generating moves...
  Generated 36 moves.
  Moving from (5,0) to (5,1).
Press [Enter] to continue.
|:
```

15 - Exemplo de uma sequência de jogadas do computador

5. Conclusão

Com este projeto concluímos que o paradigma de programação em lógica pode ser muito poderoso nas áreas de inteligência artificial desde que se tenha conhecimento do funcionamento da linguagem e do diferente modo de pensar e estruturar o código. Com este projeto, o nosso conhecimento relativo a este paradigma e, mais especificamente, à linguagem de Prolog aumentou substancialmente. No entanto, há muito mais para aprender de forma a melhorar a eficiência e de modo a adequar o código às convenções do paradigma.

6. Bibliografia

http://www.chessvariants.com/invention/latrunculi-petteia-xxi

https://sicstus.sics.se/sicstus/docs/4.3.2/html/sicstus/

7. Anexos

O código fonte encontra-se na pasta *src* em anexo.

De modo a poder utilizar o programa no SICstus Prolog é necessário consultá-lo com a instrução "*consult('$PATH/src/game.pl').*" em que *$PATH* deve ser substituído pelo diretório onde este relatório se encontra. Depois disto, o programa encontra-se pronto a correr com "*main.*".