

Latrunculi XXI

Relatório Intercalar

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, 10 de outubro de 2017

Índice

[1. **Descrição do jogo** 3](#_Toc495838031)

[a. História 3](#_Toc495838032)

[b. Regras 4](#_Toc495838033)

[c. Regras Específicas 6](#_Toc495838034)

[2. **Representação do Estado do Jogo** 9](#_Toc495838035)

[3. **Visualização do Tabuleiro em Modo de Texto** 1](#_Toc495838036)1

[4. **Movimentos** 1](#_Toc495838039)2

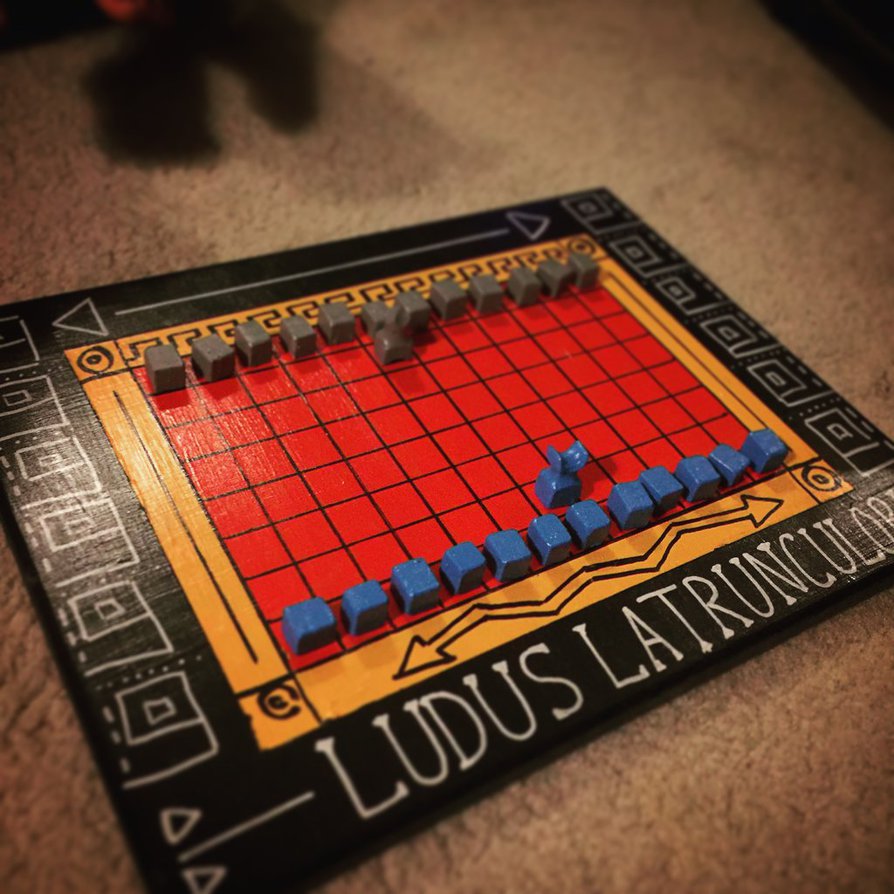
1. Descrição do jogo

Latrunculli XXI é um jogo de estratégia, que pode ser jogado com um tabuleiro 8x8, 10x8 e 12x8. Cada jogador tem na sua posse 8, 10 ou 12, dependendo do tamanho do tabuleiro, peças chamadas “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.

* 1. 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.

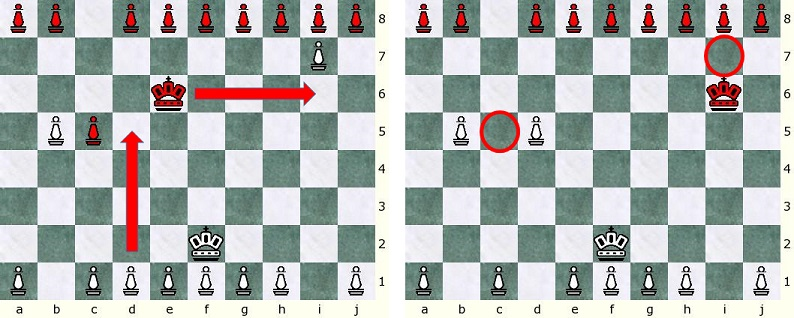
1 - Latrunculi XXI

* 1. 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.

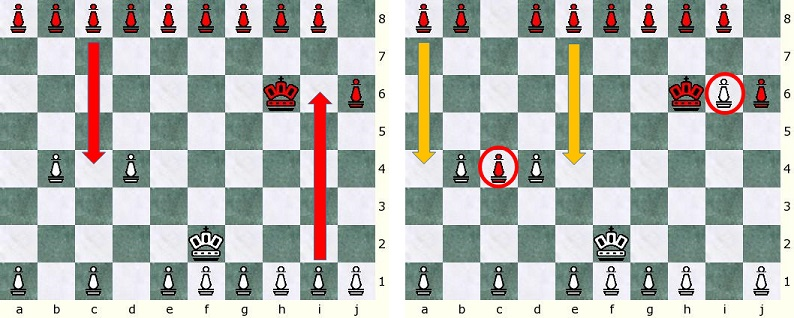
As peças movem-se ortogonalmente, ou seja, não são permitidas jogadas diagonais. Podem deslocar-se um número arbitrário de células, porém não podem saltar outras peças.

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



2 - Exemplo de Captura

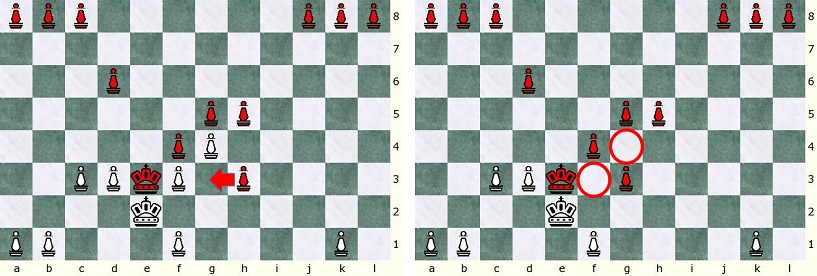
No entanto, é perfeitamente seguro uma peça mover-se para o meio de duas peças inimigas, não estando sujeitas a captura imediata (ou suicídio).



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.

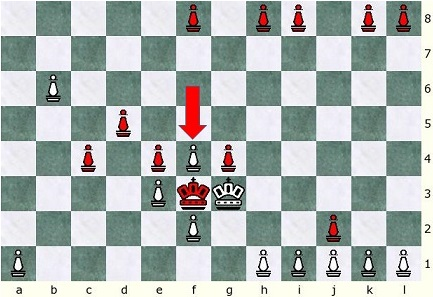
É também possível capturar múltiplas peças inimigas numa jogada.



4 - Exemplo de captura múltipla

Uma exceção à regra acontece quando a peça a capturar se encontra num dos cantos do tabuleiro. Neste caso, continuam a ser necessárias 2 peças a rodeá-la, no entanto cada uma deve bloquear uma célula adjacente à inimiga.

O objetivo do jogo é imobilizar o dux do oponente. Se ocorrer um impasse (um jogador não tem jogadas legais para fazer), o jogador em questão perde o jogo assim como quem perde todos os soldados. 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

* 1. Regras Específicas

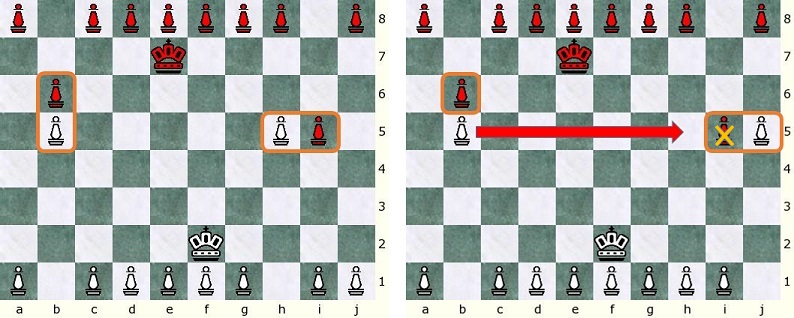
Em adição às regras tradicionais, a versão Latrunculi XXI, possui um conjunto adicional de regras.

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.

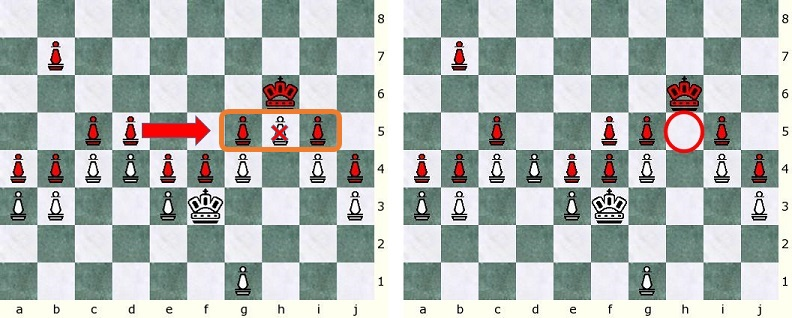
Chama-se conflito a uma situação em que uma peça está adjacente a uma inimiga. Um grupo de 3 ou mais peças envolvidas num conflito chama-se uma disputa.

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.

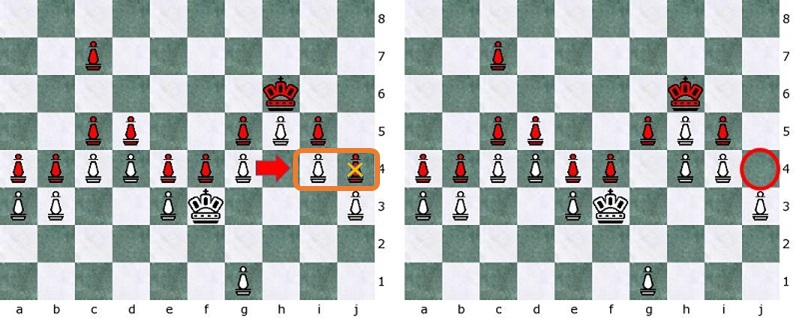
6 - Exemplo de um soldado bloqueado, mas movendo-se para fazer uma jogada ofensiva



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.

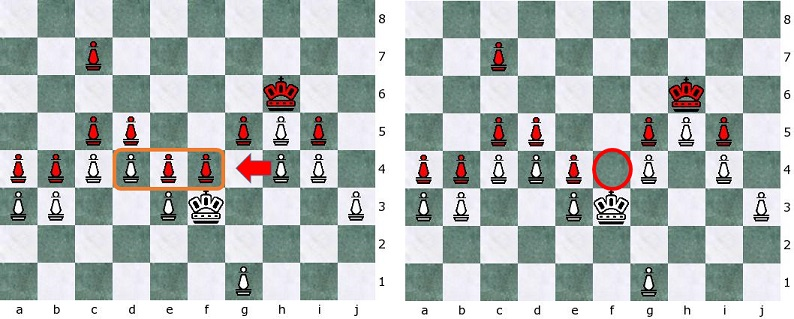


8 - Exemplo da técnica contra uma peça amiga



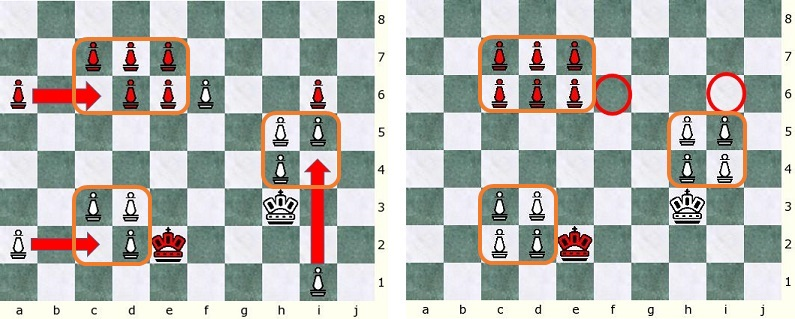
7 - Exemplo da técnica contra as bordas

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 soldados, 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:

|  |  |  |  |
| --- | --- | --- | --- |
| Posição  do Dux  Direções  bloqueadas | 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

1. Representação do Estado do Jogo

É muito importante para o jogo funcionar corretamente termos o estado do jogo bem organizadas e de uma forma não muito complexa, por isso pensamos nos principais pontos:

Tabuleiro atual – Este representa a posição de todas as peças no estado atual, será utilizada uma representação através de listas de listas, em que cada elemento representa uma célula do tabuleiro, com o conteúdo dessa célula, recorrendo a átomos.

initialBoard([

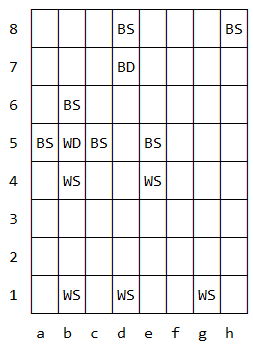
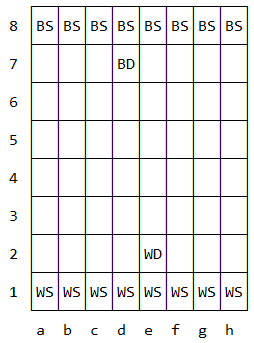
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [black\_soldier, | black\_soldier, | black\_soldier, | black\_soldier, | black\_soldier, | black\_soldier, | black\_soldier, | black\_soldier], |
| [empty\_cell, | empty\_cell, | empty\_cell, | black\_dux, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell], |
| [empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell], |
| [empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell], |
| [empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell], |
| [empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell], |
| [empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | white\_dux, | empty\_cell, | empty\_cell, | empty\_cell], |
| [white\_soldier, | white\_soldier, | white\_soldier, | white\_soldier, | white\_soldier, | white\_soldier, | white\_soldier, | white\_soldier]]). |

12 – Representação do estado inicial do tabuleiro

finalBoard([

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| [empty\_cell, | empty\_cell, | empty\_cell, | black\_soldier, | empty\_cell, | empty\_cell, | empty\_cell, | black\_soldier], |
| [empty\_cell, | empty\_cell, | empty\_cell, | black\_dux, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell], |
| [empty\_cell, | black\_soldier, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell], |
| [black\_soldier, | white\_dux, | black\_soldier, | empty\_cell, | black\_soldier, | empty\_cell, | empty\_cell, | empty\_cell], |
| [empty\_cell, | white\_soldier, | empty\_cell, | empty\_cell, | white\_soldier, | empty\_cell, | empty\_cell, | empty\_cell], |
| [empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell], |
| [empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell, | empty\_cell], |
| [empty\_cell, | white\_soldier, | empty\_cell, | white\_soldier, | empty\_cell, | empty\_cell, | white\_soldier, | empty\_cell]]). |

13 – Representação do estado final do tabuleiro – Derrota do dux branco



15 - Visualização do estado final

14 - Visualização do estado inicial

Estado da Peça – Como visto nas regras existem momentos em que, por exemplo, as peças podem estar bloqueadas e não se conseguem mover, por isso é necessário ter essa informação para cada peça.

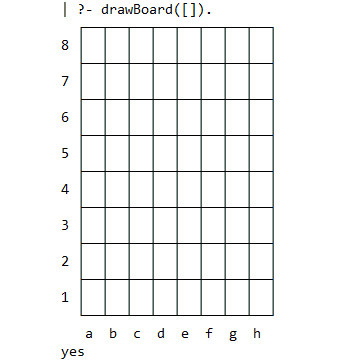
Atualizar estado do jogo – Após cada jogada é necessário atualizar o estado do jogo, ou seja, é preciso verificar se a jogada feita captura ou bloqueia alguma peça do adversário de modo a atualizar os estados das peças.

Verificar se soldado foi capturado – Isto é necessário quando feita a atualização do estado do jogo, para se excluir a peça do tabuleiro.

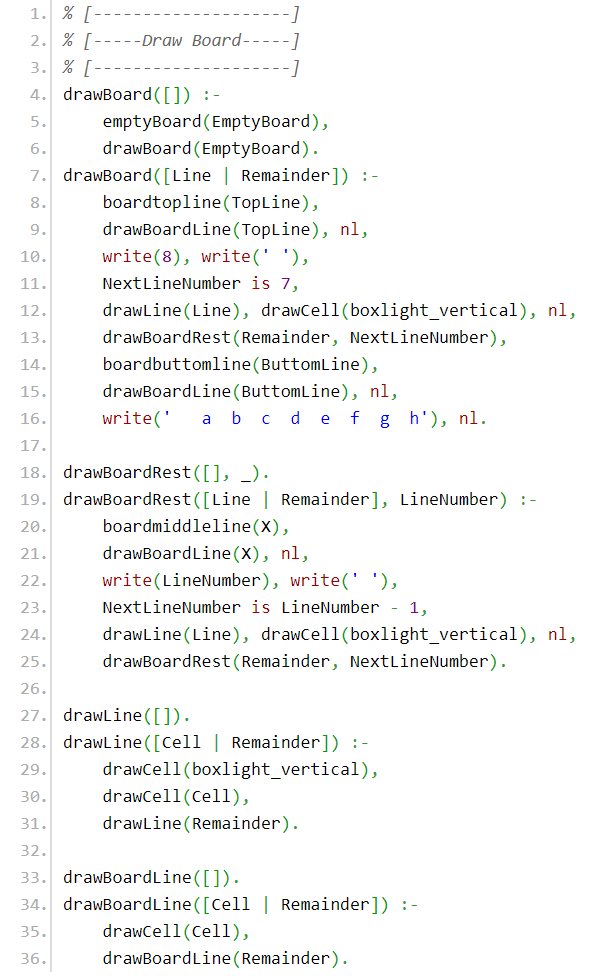
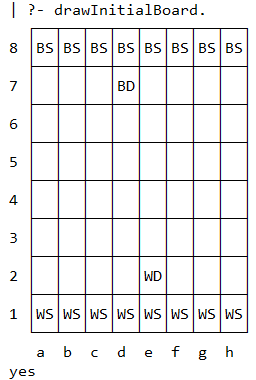
Verificar fim de jogo – Ao decorrer de cada jogada é necessário verificar se o jogo acabou, ou seja, verificar se o adversário ainda tem possibilidades de deslocar alguma peça, se ainda contém peças no tabuleiro ou se o Dux foi capturado.

1. Visualização do Tabuleiro em Modo de Texto

Para a visualização do tabuleiro em qualquer altura do jogo, estamos a usar 5 átomos diferentes, “black\_soldier”, “black\_dux”, “white\_soldier”, “white\_dux” e “empty\_cell”. Deste modo drawBoard() recebe listas de listas que depois é processado para fazer a representação correta do jogo, quando lhe é passado uma lista vazia ele apenas desenha o tabuleiro sem peça nenhuma. Para melhores efeitos visuais é recomendado alterar a fonte da consola para “Consolas”.



17 - Output do tabuleiro vazio



18 - Output do tabuleiro inicial

16 - Código usado para visualizar o tabuleiro na consola

1. Movimentos:

Selecionar peça – Modo de obter uma peça do tabuleiro.

selectPiece(Column, Row, Board, Piece).

Verificar jogada – Verifica se a jogada é válida.

checkMove(FromColumn, FromRow, ToColumn, ToRow, Board).

Este predicado fará uso de outros como:

isElementBetween(Board, Xi, Yi, Xf, Yf). – Verifica se existem peças entre dois quadrados do tabuleiro.

isOrthogonal(Xi, Yi, Xf, Yf). – Verifica se a jogada é ortogonal.

isEmptyCell(empty\_cell). – Verdadeiro apenas se a célula estiver vazia.

Mover peça – Mover a peça para uma posição pretendida, de modo a apagar a posição em que ela estava antes e coloca-la no tabuleiro na nova posição.

move(FromColumn, FromRow, ToColumn, ToRow, Board, NewBoard).

Processamento posterior – Processa a jogada de modo a verificar se houve captura(s) e se o jogo chegou ao fim.

postProcessMove(FromColumn, FromRow, ToColumn, ToRow, Board).