Asterismo

Relatório Final



Mestrado Integrado em Engenharia Informática e de Computação

Programação em Lógica

Grupo Asterismo 2:

José António Barbosa Fonseca Guerra – up201706421

Martim Pinto da Silva - up201705205@fe.up.pt

Faculdade de Engenharia da Universidade do Porto

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

14 de Novembro de 2019

**Resumo**

Este projeto consiste na conceção de um jogo de tabuleiro chamado Asterismo realizado através de um conjunto de regras utilizando a linguagem de programação denominada Prolog.

O jogo escolhido foi um jogo cooperativo de 2 jogadores havendo a possibilidade de se jogar com dois jogadores reais, 1 jogador real e 1 jogador virtual ou 2 jogadores virtuais (2 máquinas). Cada um dos jogadores virtuais tem diferentes níveis de dificuldade (AI) até um nível máximo de 2.

Todas as regras e modos foram implementadas com sucesso para a entrega deste projeto.

Este projeto permitiu uma aprendizagem de forma eficaz para a realização do trabalho, baseando-se muito na pesquisa e na motivação pela parte prática da linguagem de Prolog.

O que se pôde verificar, logo desde o início da utilização desta ferramenta, foi a rapidez e eficácia da realização de qualquer tarefa, nomeadamente problemas de decisão.

Resumindo, foi realizado um jogo complexo numa linguagem nova e completamente diferente do que estávamos habituados mas que há medida do tempo, foi se tornando mais desafiadora e interessante até á final da realização do trabalho. Gostaríamos de ter melhorado ainda mais a parte de interface gráfica, mas sabemos que esse trabalho será desenvolvido mais na unidade curricular de LAIG.

# Índice

* Introdução…………………………………………….4
* Jogo Asterismo………………………………………..5
* Lógica do Jogo………………………………………..8
  + Representação interna do estado do jogo ….....8
  + Visualização do tabuleiro …………………….
  + Lista de jogadas Válidas ……………………..
  + Execução de jogadas………………………….
  + Final do Jogo……………………………………
  + Avaliação do Tabuleiro…………………………….
  + Jogada do Computador……………………………
* Conclusões……………………………………………
* Bibliografia……………………………………………

# Introdução

Este projeto foi desenvolvido no âmbito da unidade curricular de Programação em Logica (PLOG), do 3º ano do curso Mestrado Integrado em Engenharia Informática da FEUP. O objetivo deste projeto foi implementar, através do sistema de desenvolvimento do SICSTUS Prolog, um jogo de tabuleiro com um conjunto de regras pré-definidas pelo criador do jogo.

O relatório segue a seguinte estrutura:

* Jogo Asterismo: Descrição do jogo e principalmente das suas regras.
* Lógica do Jogo: Implementação da lógica do jogo em Prolog, tendo a seguinte estrutura:
  + Representação interna do estado do jogo: Descrição da estrutura utilizada para guardar o estado inicial, intermédio e final do jogo
  + Visualização do tabuleiro: Exposição da interface com o utilizador, descrevendo a visualização do estado do jogo e robustez da interface/validações de entrada.
  + Lista de jogadas válidas: Descrição dos predicados utilizados para a obtenção das jogadas válidas.
  + Execução de jogadas: Exposição dos predicados utilizados para fazer com que o ciclo do jogo funcione.
  + Final do jogo: Descrição dos predicados que fazem a verificação do final do jogo com identificação do vencedor.
  + Avaliação do tabuleiro: Descrição dos predicados que permitem comparar a aplicação de diversas jogadas disponíveis.
  + Jogada do computador: Descrição dos predicados que fazem com que o computador escolha uma jogada válida durante o ciclo de jogo bem como a explicação da implementação das dificuldades 0 e 1 do computador.
* Conclusões: Conclusões do trabalho e opiniões acerca do que poderia ter sido feito para melhorar o trabalho desenvolvido.
* Web grafia: Lista de sites utilizados para a realização do trabalho.

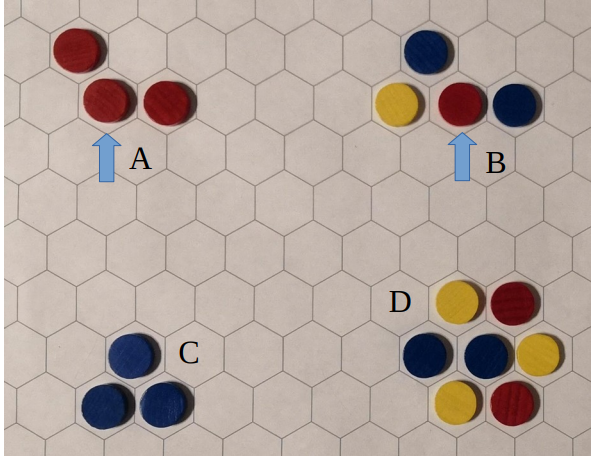
# Jogo Asterismo

# 

Este jogo de tabuleiro foi criado em 2019 por Giuliano Polverari.

É um **jogo cooperativo** para ser jogado entre 2 pessoas num tabuleiro com peças hexagonais. Há 63 peças de 3 cores diferentes (vermelho, amarelo e azul).

O objetivo principal dos jogadores é obter 5 peças de cada cor cada um, **sem perturbar o equilíbrio da arvore de jogo.**



**Figura 2**: Explicação

Regras:

Qualquer peça pode ser retirada, no entanto as peças adjacentes têm de estar **seguras**.

Uma peça está **segura** se estiver ligada a duas peças da mesma cor (situação A) ou ligada a 3 peças de qualquer cor (situação B).

Cada jogador por turno faz apenas uma jogada, ou seja remove 1 peça, se possível.

Cada jogador tem um limite máximo de 5 peças para cada cor.

Ao retirarem peças, os jogadores têm de verificar se **não quebram a árvore**, isto é ao retirar um peça do tabuleiro não fique com dois ou mais conjuntos de peças separados uns dos outros.

Se os 2 jogadores chegarem as 5 peças de cada cor vencem “o jogo”, senão “o jogo” vence e os jogadores saem derrotados.

Prémios:

* 2º lugar Cogita 2019 ("melhor jogo abstrato de estratégia”) 5ª edição.

**Figura 1**: tabuleiro do jogo



Situação inicial:

As peças são colocadas aleatoriamente pelo tabuleiro.

As peças, que devido a configuração inicial não estejam **seguras**, terão de ser atribuídas a cada jogador da forma aleatória em igual número.

**Figura 3**: Situação inicial

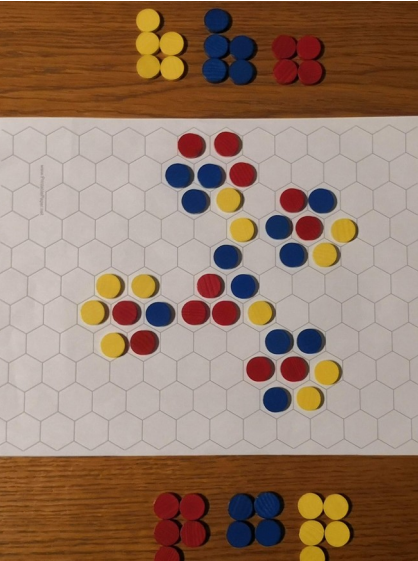


Desenvolvimento:

Vai-se retirando e colocando peças, respeitando as regras de jogo, tendo como objetivo principal acumular 5 peças de cada cor.

Os 2 jogadores jogam cooperativamente, ou seja, têm de acumular as dois 5 peças de cada cor para alcançar a vitória.

O jogador terá de ter especial atenção para não quebrar a arvore do jogo, isto é não poderá haver blocos sem peças adjacentes.



**Figura 5**: Situação final

Situação final:

O jogador terá de estar atento a situações em que possui 14 peças, assim como o seu colega de equipa.

Nestas situações, é provável que não seja possível retirar mais peças sem quebrar a árvore do jogo. Nesta condição perdem os jogadores e ganha “o jogo”, tal como neste exemplo. Se por outro lado for possível cada 1 alcançar as 5 peças de cada cor os 2 jogadores vencem “o jogo”.

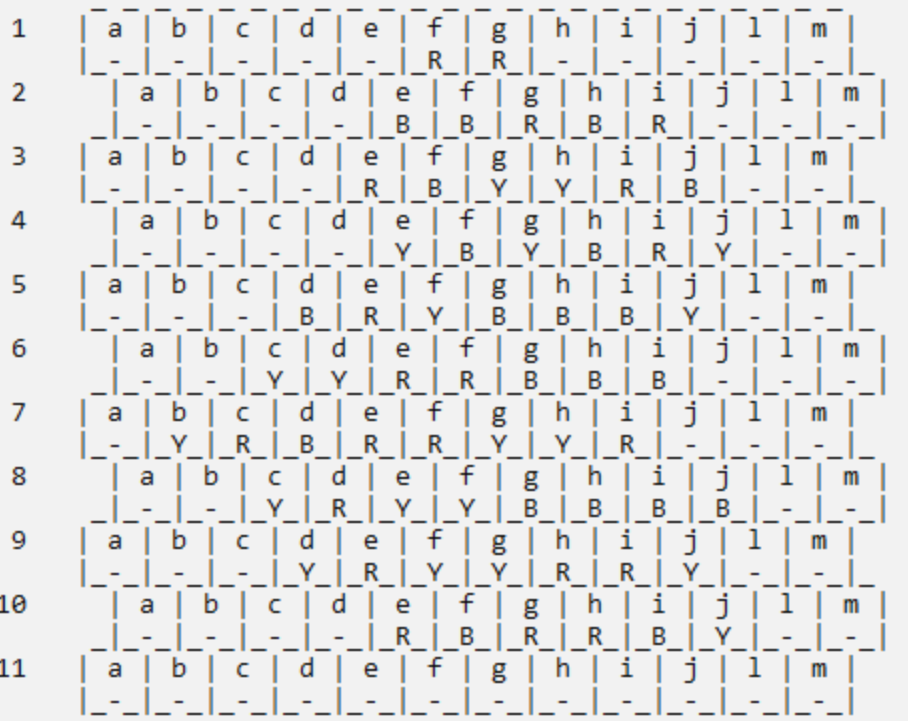
**Figura 4**: Situação desenvolvimento

# Lógica do Jogo

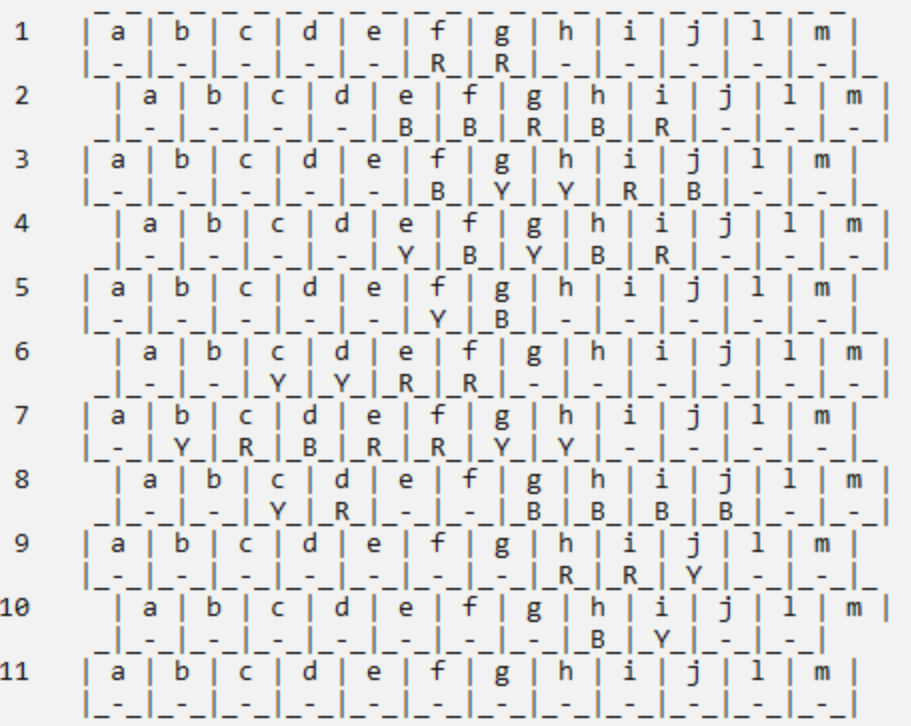
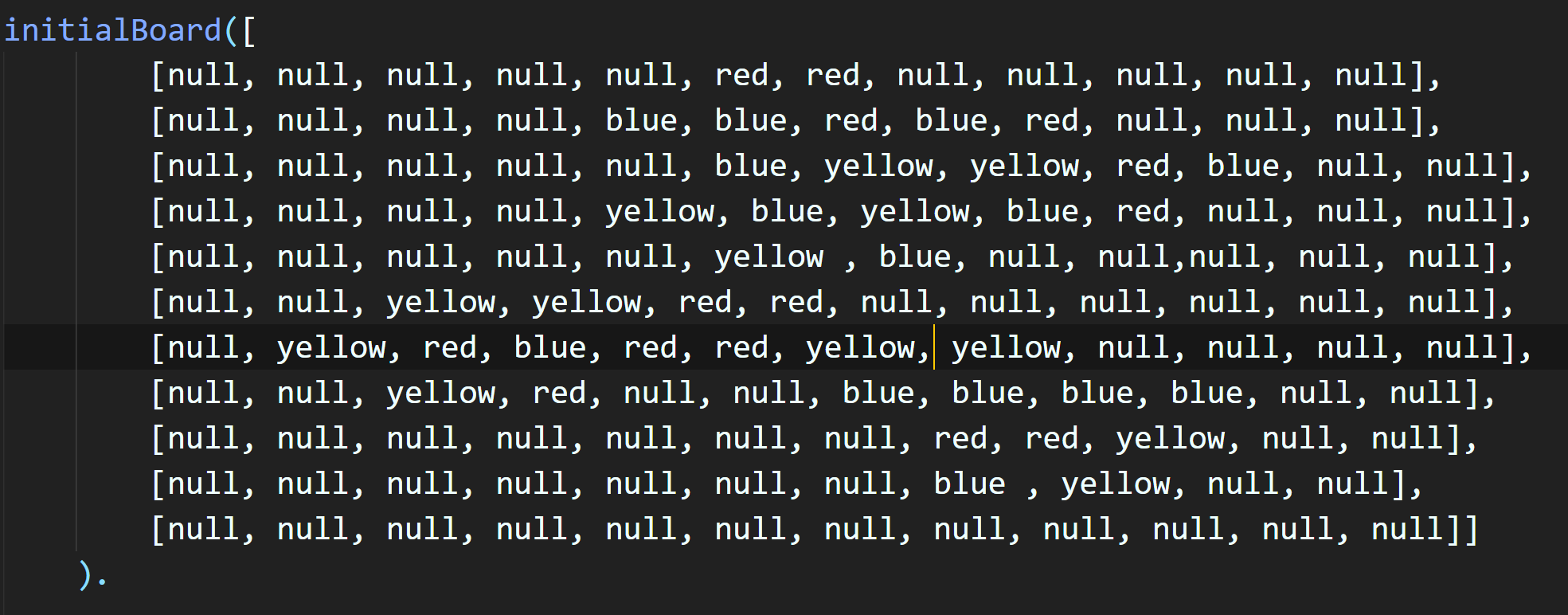
# Representação interno do estado do jogo

# Situação inicial:

# 

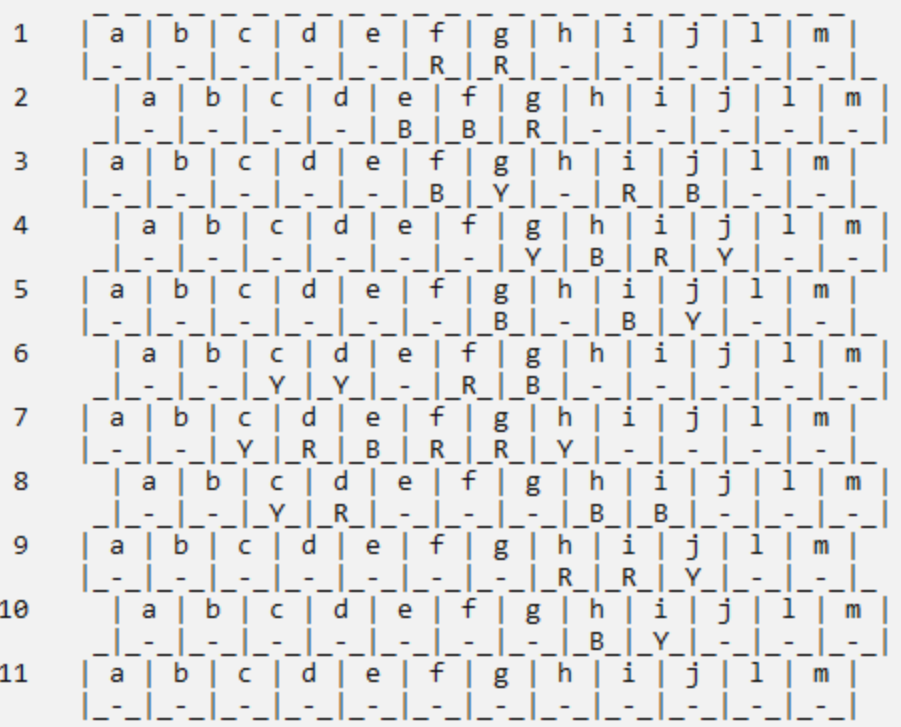


# Desenvolvimento:



# Situação final:

# 



# Visualização do tabuleiro – atualizar print

# 

Lista de jogadas válidas

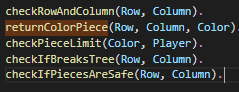
Para uma jogada ser válida é necessária que em primeiro lugar o jogador ao escolher uma posição do tabuleiro esta corresponda a uma peça das 3 cores disponíveis e não uma posição vazia, pelo que é necessária que seja feita uma verificação dos valores Row (linha) e Column (coluna) que são pedidos ao jogador como input.

Para verificar se uma jogada é válida criou-se um predicado chamado **checkRules (Row, Column,Player, IsMachine),** que se encontra no ficheiro rules.pl, onde se passa a coluna e a linha do tabuleiro, variáveis Row e Column respetivamente, que são valores providenciados pelo jogador para a suposta peça que ele quer retirar. A variável Player serve para identificar o jogador em si e a variável IsMachine é utilizada sempre ou com o valor 0 ou com valor 1 e serve para indicar se é a Máquina que vai utilizar este predicado (IsMachine é unificado com o valor 1) ou um jogador humano (IsMachine é unificado com 0), isto é feito para que caso seja um jogador humano então se este escolher uma jogada inválida uma mensagem de erro apareça no ecrã, caso seja uma máquina não se quer este comportamento quer-se que o predicado checkRules falhe para que depois ele volte a tentar com um valor novo de Row e Column a seguir. Este predicado checkRules verifica se para a posição Row (linha) e Column (coluna) as regras do jogo se verificam, isto é se o jogador efetivamente pode concluir a jogada com a retirada de uma peça.

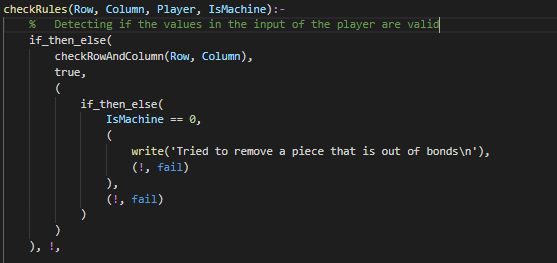


**Figura** :Predicado checkRules()

Dentro do checkRules() é necessário verificar diversas condições para que consigamos concluir se com o input do jogador consiguemos retirar uma peça , para tal utilizamos predicados dentro do checkRules() que se encontram no ficheiro game.pl (**returnColorPiece(Row, Column, Color)**), no ficheiro rules.pl ( **checkRowAndColumn(Row, Column)** , **checkIfBreaksTree(Row, Column)** , **checkIfPiecesAreSafe(Row, Column)**) e finalmente no ficheiro players.pl ( checkIfPieceLimit

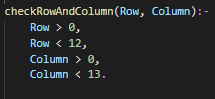


mas em primeiro lugar é necessário fazer uma verificação ao nível do input do player, isto é se os valores Row e Column que são providenciados pelo player são válidos ou não. Para isto temos um predicado chamado **checkRowAndColumn(Row, Column)** que indica se a Row e a Column se encontra dentro do espectro de opções possíveis (Row entre 1 e 11 e Column entre 1 e 12).



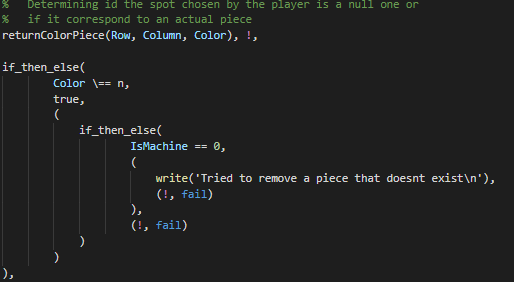
**Figura** : Chamada do predicado checkRowAndColumn()

dentro do checkRules()



**Figura** :Predicado checkRowAndColumn()

Depois desta verificação é necessário ir verificar se na posição que o jogador escolheu efetivamente se encontra uma peça ou não, para isso temos o predicado **returnColorPiece(Row, Column, Color)** ao qual se passa a Row e a Colum (Posição) e na variável Color ele retorna a cor da peça para a posição correspondente. É de notar que na representação do tabuleiro está a ser utilizada a letra n para representar uma posição onde não se encontra nenhuma peça, pelo que caso este predicado retorne n na Color quer dizer que estamos perante uma posição que não contém nenhuma peça, sendo assim o predicado checkRules vai falhar.



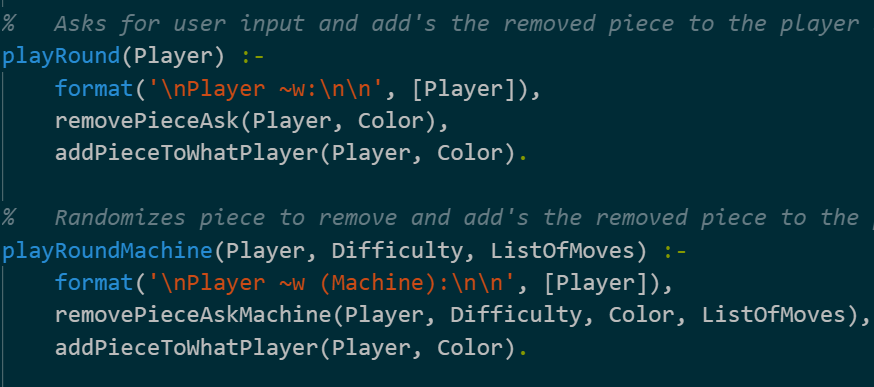
**Figura** : Chamada do predicado returnColorPiece()

dentro do checkRules()

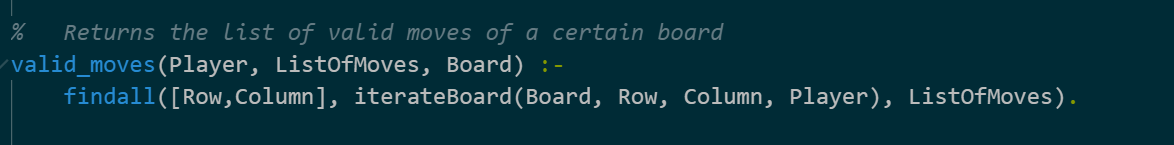
Dentro do predicado **returnColorPiece(Row, Column, Color)**

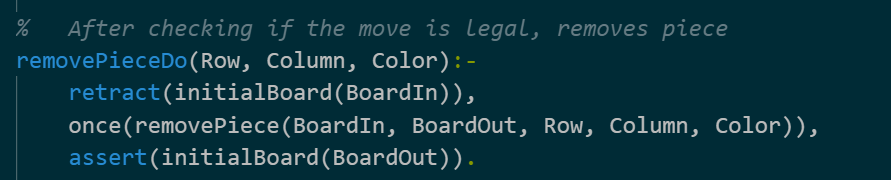
Execução de jogadas

O nosso jogo tem um ciclo principal chamado playLoop localizado no ficheiro **game.pl**

A cada jogada é chamado o predicado playRound ou playRoundMachine dependendo do modo de jogo escolhido. Os modos de jogo variam entre jogador contra jogador, jogador contra máquina e máquina contra máquina.

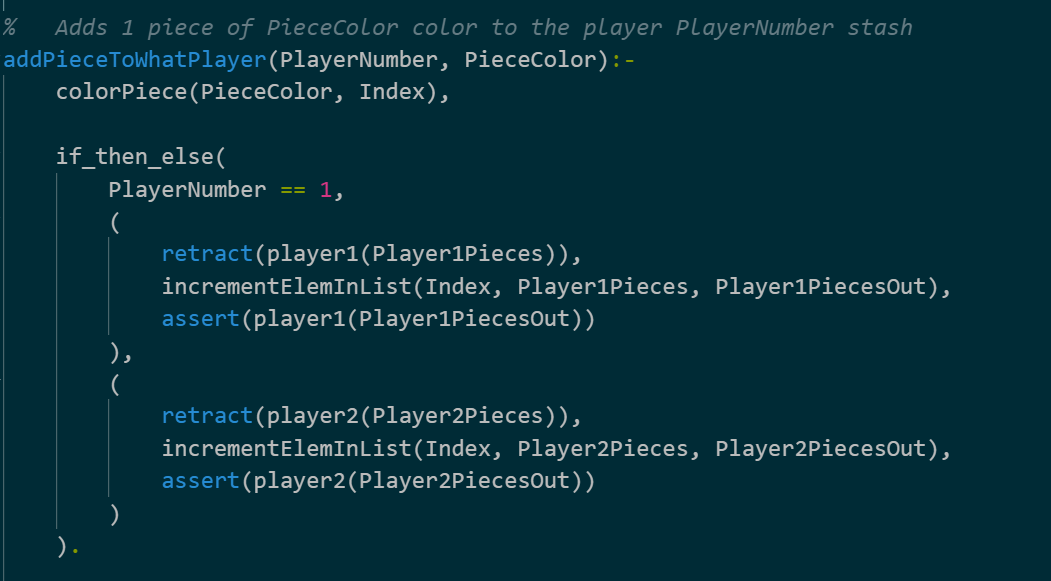
Em ambos os predicados é escolhida a peça a ser retirada (através de input do jogador ou através de geração de uma jogada por parte da máquina).

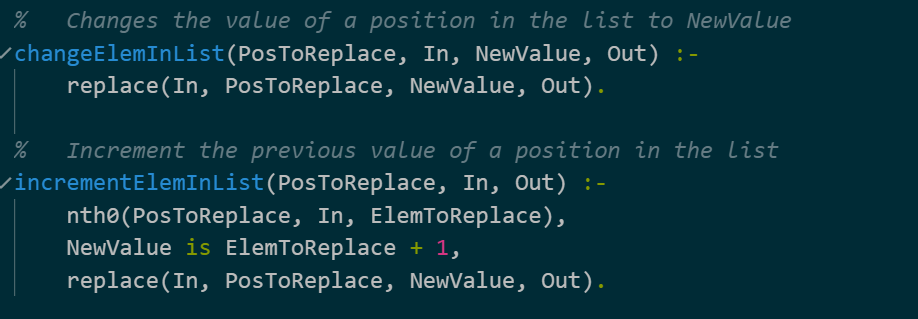
Para gerar a jogada da máquina são feitos dois níveis de “AI”. No nível 0 é feito uma geração aleatória de uma coluna e linha e se uma peça for possível ser retirada o jogo prossegue. No caso do nível 1 é gerado primeiro uma lista de jogada possíveis e escolhida aleatoriamente uma jogada dessa lista com auxílio ao predicado findall.

Ambos os predicados removePiecesAsk e removePieceAskMachine, depois de ser escolhida a peça a ser retirada verificam se, primeiro, a jogada está dentro do tabuleiro, e em segundo se todas as regras do jogo são cumpridas. Em ambos os casos, em caso de insucesso é escolhida uma nova jogada e em caso de sucesso a peça retirada é adicionada ao “inventário” de peças do jogador através do predicado removePieceDo.

Para a verificação de jogadas foi criado um ficheiro auxiliar chamado **rules.pl** que faz todo o tipo de verificações. Verifica se está alguma peça na jogada selecionada, se não há nenhuma peça, adjacente à selecionada que fique desprotegida e se a arvore do jogo não é partida (consultar regras no início).

De seguida, é adicionada a peça ao jogador na chamada addPieceToWhatPlayer, na qual é incrementada uma das peças (dependendo da cor da peça selecionada) do inventário do jogador.



Para isso optámos por escolher duas funções de fácil uso nas quais é feito a mudança de um elemento qualquer da lista através de predicados de uso complexo. Este conjunto de predicados, como é partilhado por diversos ficheiros está num documento á parte chamado **shared.pl**

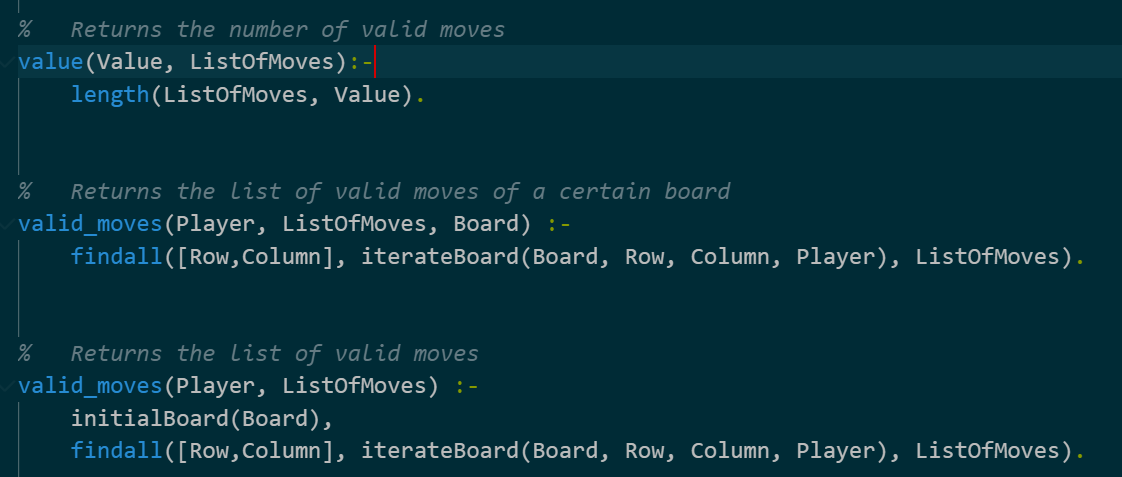
Por último, como cada jogador tem obrigatoriamente de retirar uma peça, é verificado, no final do jogo, se os dois jogadores ganharam, ou seja, se no final das duas jogadas têm um total de 10 peças de cada cor, tendo este valor ter de ser distribuído igualmente (5 para cada).

É também verificado após cada jogada, se o jogo ganhou, ou seja, se não há mais nenhuma peça possível de ser retirada sem que os jogadores obtenham o total necessário para alcançar a vitoria (Não esquecer que este jogo é cooperativo).

Avaliação do Tabuleiro

Os 3 predicados mais importantes na avaliação do tabuleiro são os descritos em baixo.

O predicado value avalia um tabuleiro retornando o número possível de jogadas nesse tabuleiro. O predicado valid\_moves cria um tabuleiro com todas as jogadas possíveis através da chamada ao predicado findall



Conclusões

O projeto teve como objetivo aplicar o conhecimento adquirido sobre a linguagem Prolog de forma iterativa e desafiadora.

Ao longo do desenvolvimento deste trabalho, encontramos diversos problemas, tais como, usar nos sítios corretos a grande vantagem que é o backtracking e usar efetivamente os cuts. Tentámos também preocuparmo-nos mais com a eficiência (poderíamos tê-lo feito por exemplo usando o predicado inline) mas no final não deu tempo para tudo.

Em suma, foi uma experiência positiva, na qual aprendemos muito mais do que o que foi dado nas aulas teóricas devido ao nosso interesse e dever de pesquisar assuntos não abordados ou não compreendidos totalmente.