**Jogo de tabuleiro “Mod X”**

Relatório Intercalar



Mestrado Integrado em Engenharia Informática e Computação

Programação em Lógica

Grupo “Mod X” 4:

André Sousa Lago – 201303313

Gustavo Rocha da Silva – 201304143

Faculdade de Engenharia da Universidade do Porto

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

6 de Outubro de 2015

# 1 – Introdução

Este trabalho foi desenvolvido no âmbito da unidade curricular “Programação em Lógica” do 3º ano do Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto. O seu objetivo é o de implementar em Prolog um jogo de tabuleiro de 2 jogadores de forma a possibilitar o jogo Humano vs. Humano, Humano vs. Computador e Computador vs. Computador.

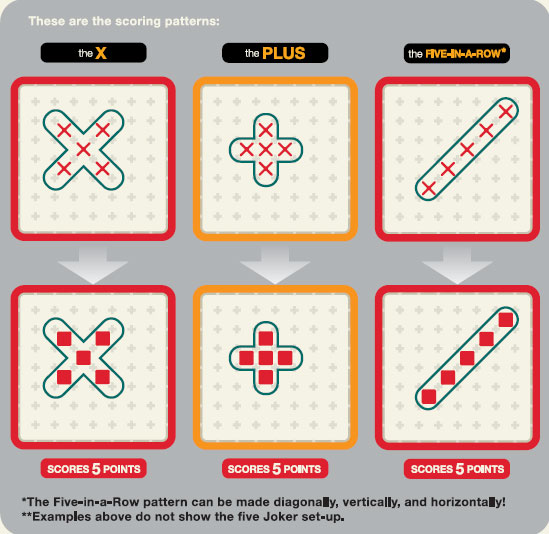
Neste relatório será descrito o jogo que escolhemos para a nossa implementação – o “Mod X” – bem como as suas regras. De seguida, serão detalhadas algumas funcionalidades e características da nossa implementação, desde a representação do jogo e visualização do tabuleiro até a avaliação de jogadas pelo computador e final do jogo. Por fim, serão apresentadas as conclusões que obtivemos da realização deste trabalho, bem como a sua bibliografia.

# 2 – O Jogo “*Mod X*”



O “*Mod X*” é um jogo de tabuleiro criado pela “*Cryptozoic Entertainment*”, lançado em Dezembro de 2014, pertencendo à gama de estratégia para 2 a 4 jogadores de idades superiores a 14 anos. Cada jogo possui uma duração aproximada de 25 minutos.

O jogo é jogado num tabuleiro dividido em células quadradas (8x8) e tem um funcionamento bastante simples. A cada jogador é atribuída uma cor (vermelho, preto, amarelo e laranja), bem como 14 peças-X e 18 marcadores de pontuação (ou *joker*’s) da respetiva cor a cada um. O objetivo do jogo é que cada jogador coloque, à vez, uma peça-X no tabuleiro de forma a criar padrões de peças-X (“X”, “+” ou “5 em linha”). Uma vez criado um padrão, o jogador coloca marcadores de pontuação nas posições onde tinha peças-X, retirando estas para que possam ser reutilizadas. Um jogador que construa um padrão sobre os marcadores já posicionados pode sobrepor os existentes com marcadores seus, reduzindo a pontuação do seu adversário. Em cada momento, a pontuação de um jogador corresponde ao número de células do tabuleiro cujo marcador de cima é seu, sendo que o jogo termina quando um dos jogadores atinge um valor de pontuação limite pré-estabelecido ou um jogador esgota todas as peças que tem para jogar (neste caso, ganha o jogador com mais pontos).

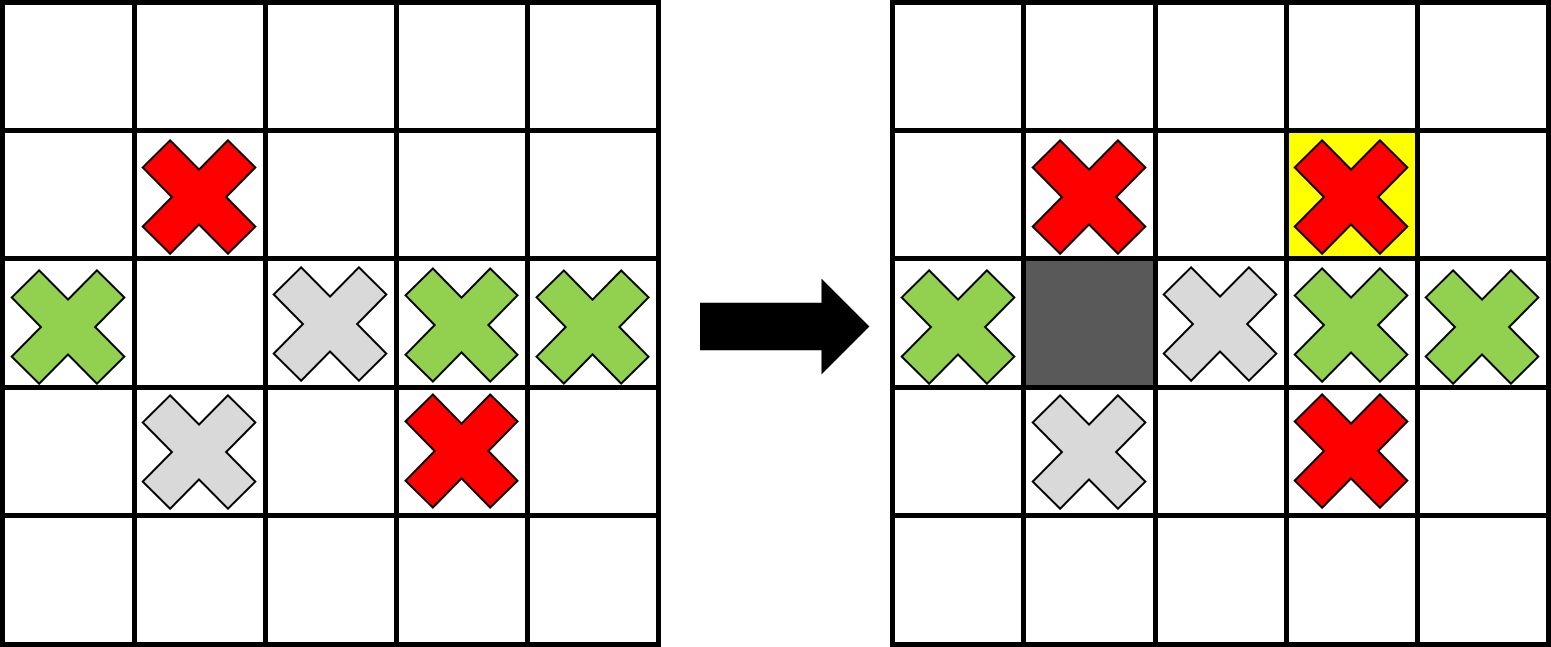


Um padrão tem de ser constituído apenas por peças-X da cor do jogador e por peças-X transparentes. No início do jogo 5 destas peças transparentes são colocadas de forma aleatória no tabuleiro para que possam ser utilizadas por qualquer jogador. Se um jogador conseguir fazer um padrão utilizando uma ou mais peças transparentes, deve substituí-las por marcadores de pontuação da sua cor e reposicioná-las no tabuleiro desde que a nova posição não dê imediatamente origem a um novo padrão.

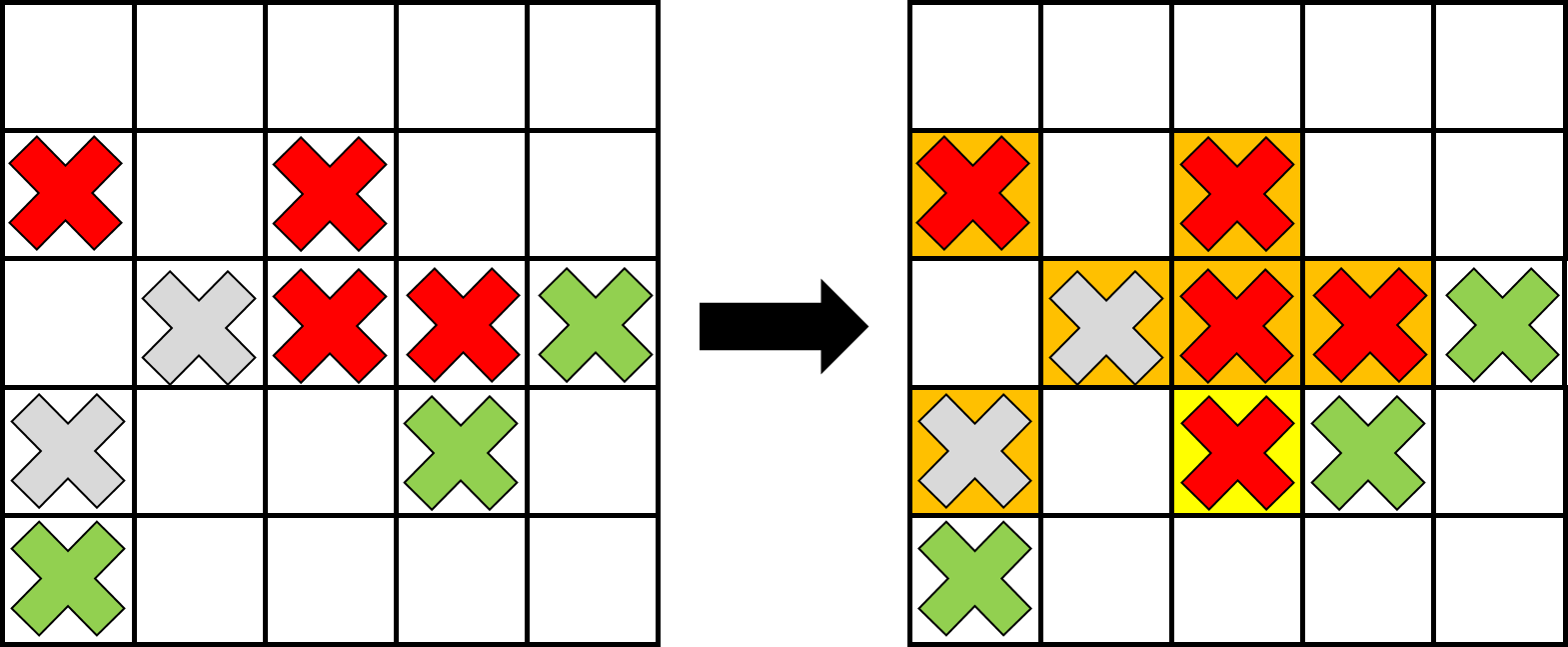
Padrões possíveis a realizar. O padrão “5 em linha” é também possível na horizontal ou vertical.

Seguem-se alguns exemplos de situações particulares do jogo:

Na sequência à esquerda, o jogador de vermelho pontua 5 pontos por colocar uma peça na posição assinalada a amarelo. Assim, se fizer isto, retira as três peças vermelhas bem como as duas transparentes uma vez que as usou no seu padrão, colocando de seguida os seus marcadores de posição. Por fim, deve reposicionar as peças transparentes em qualquer posição à exceção da que está assinalada a cinzento pois isso causaria o surgimento de um padrão “5 em linha verde”.



Na imagem à direita pode observar-se outra situação possível no jogo: a de se fazerem dois padrões em simultâneo. Do quadro da esquerda para o da direita, se o jogador de vermelho colocar a sua peça na posição assinalada a amarelo consegue obter dois padrões: uma cruz (X) e um “mais” (+) com recurso a 4 outras peças vermelhas e a 2 peças transparentes. Deste modo, com uma só jogada o jogador obtém 7 pontos.



# 3 – Lógica do jogo

## 3.1 – Representação do Estado do Jogo

No caso da nossa implementação do jogo o número de peças vai ser um pouco diferente uma vez que apenas é possível jogar com 2 jogadores (humano-humano, humano-computador, computador-computador).

Cada jogo “Game” é representado como uma lista com 4 elementos. O primeiro “Board” representa o estado do tabuleiro e o segundo, “Player”, representa o jogador que deve fazer a jogada seguinte. O terceiro e quarto elemento representam, respetivamente, as informações sobre o jogador 1 e o jogador 2.

A representação do estado do tabuleiro é feita com recurso a uma lista de linhas do tabuleiro (em que a primeira corresponde à linha de cima). Por sua vez, cada linha é constituída por uma lista de células do tabuleiro (em que a primeira corresponde à célula mais à esquerda). Por fim, cada célula é constituída por uma lista de dois elementos. O primeiro é uma lista de tamanho variável que funciona como uma pilha dos marcadores de posição já existentes nessa posição. Ou seja, se o jogador vermelho tiver colocado numa posição um marcador seu (“r”) e o jogador verde tiver sobreposto este com um marcador seu (“g”), o primeiro elemento da lista dessa célula seria [g, r]. O segundo elemento é o número indicativo da peça-X presente nessa célula (a correspondência entre códigos e peças está representada na secção “Visualização do Tabuleiro”).

Os únicos requisitos impostos por esta implementação são a necessidade de todas as listas que representam linhas terem o mesmo tamanho, que deverá também corresponder ao número total de linhas (elementos da lista principal). Para além disso, os valores de ocupação de cada célula devem corresponder aos valores enunciados para que a representação seja a mais correta.

## 3.2 – Visualização do Tabuleiro

O tabuleiro será visualizado através da utilização de caracteres ASCII para representar as peças, posições vazias e limites das células. Segue-se a correspondência entre códigos e valores para as peças-X e marcadores de pontuação:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Peças-X** | | **Marcador de Pontuação** | |
|  | **Código** | **Caractere** | **Código** | **Caractere** |
| **Sem peça** | -1 | ‘ ‘ | -1 | ‘ ‘ |
| **Transparente** | 0 | ‘X’ | Inexistente | |
| **Vermelho** | 1 | ‘\*’ | 1 | ‘#’ |
| **Verde** | 2 | ‘»’ | 2 | ‘=’ |

Para além dos valores indicados na tabela, existem também caracteres usados na separação horizontal (‘|’) e vertical (‘-‘) de células.

Como exemplo da representação referida na tabela acima, segue-se uma comparação entre o estado do tabuleiro de jogo e a sua representação em texto na consola de Prolog.

0 1 2 3 4 5 6 7

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | | X X | | | |

0 | | | | | X | | | | 0

| | | | | X X | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | ##### | | | | | | |

1 | | ##### | | | | | | | 1

| | ##### | | | | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | ##### | | | | X X | |

2 | | | ##### | | | | X | | 2

| | | ##### | | | | X X | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | ##### | | | | |

3 | | | | ##### | | | | | 3

| | | | ##### | | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | | ##### | | | X X |

4 | | | | | ##### | | | X | 4

| | | | | ##### | | | X X |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | ===== | ===== | ===== | ===== | »===» | |

5 | | | ===== | ===== | ===== | ===== | ==»== | | 5

| | | ===== | ===== | ===== | ===== | »===» | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| » » | X X | \* \* | \* \* | X X | » » | \* \* | |

6 | » | X | \* | \* | X | » | \* | | 6

| » » | X X | \* \* | \* \* | X X | » » | \* \* | |

+-------+-------+-------+-------+-------+-------+-------+-------+

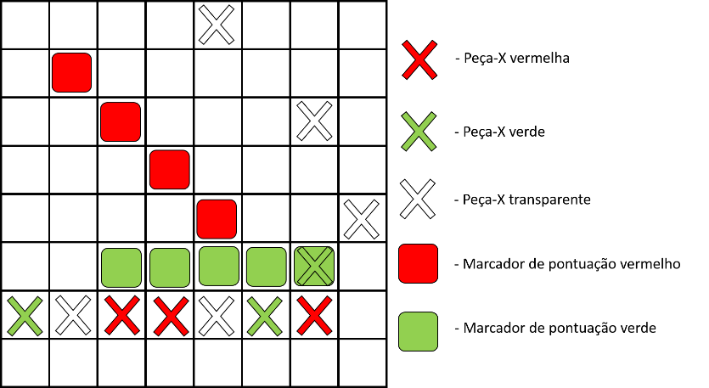
| | | | | | | | |

7 | | | | | | | | | 7

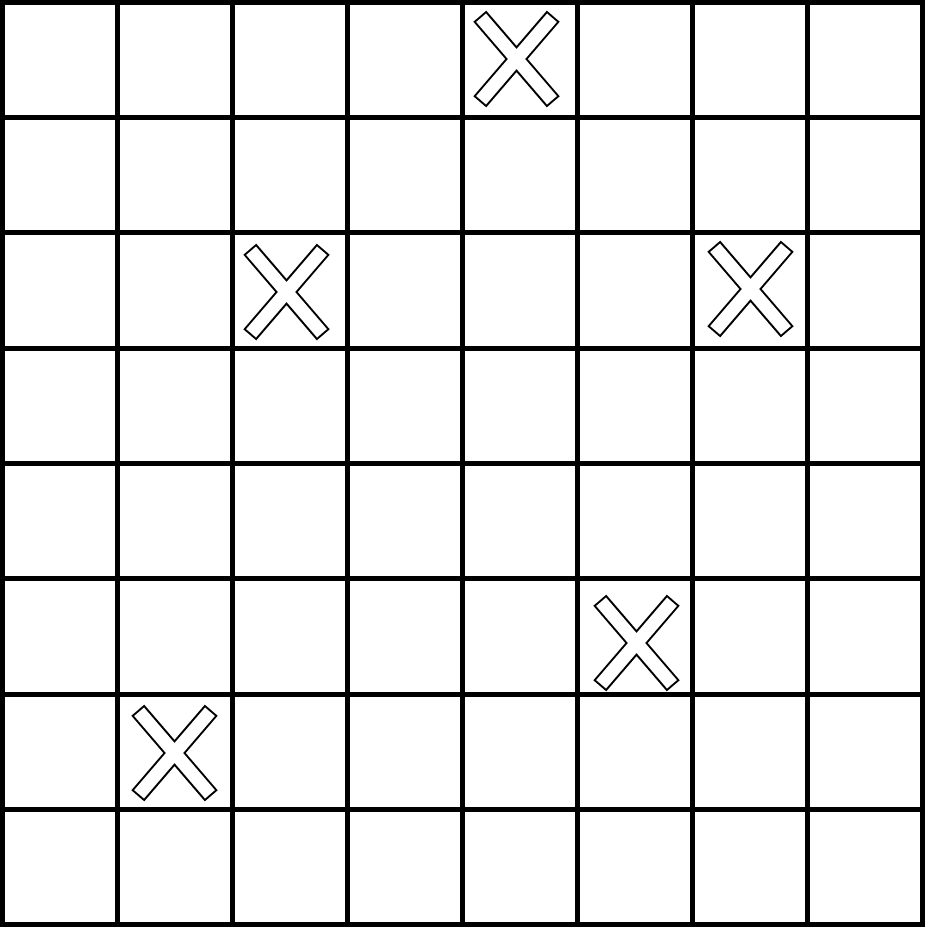
| | | | | | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

0 1 2 3 4 5 6 7



Como se pode ver, as peças-X são representadas colocando o caracter correspondente numa formação em “X”, enquanto os marcadores de posição são representados por quadrados 5x5 preenchidos com o caractere que lhes corresponde. Na coluna 7 da linha 6 é visível o resultado de se colocar uma peça-X sobre um marcador de pontuação. Por sua vez, a lista Prolog equivalente a este tabuleiro é:

O estado anterior corresponde a um estado intermédio do jogo, ou seja, a um estado em que o jogo ainda está a decorrer uma vez que todos os jogadores têm peças para jogar e nenhum atingiu a pontuação limite. Assim, segue-se a representação de um estado inicial em que apenas estão colocadas as peças-X transparentes (de notar que a posição de cada uma destas peças é aleatória) e um estado final em que o jogador verde atingiu a pontuação limite (8 pontos). Ambas as representações representam o mesmo jogo:

0 1 2 3 4 5 6 7

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | | X X | | | |

0 | | | | | X | | | | 0

| | | | | X X | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | | | | | |

1 | | | | | | | | | 1

| | | | | | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | X X | | | | X X | |

2 | | | X | | | | X | | 2

| | | X X | | | | X X | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | | | | | |

3 | | | | | | | | | 3

| | | | | | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | | | | | |

4 | | | | | | | | | 4

| | | | | | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | | | X X | | |

5 | | | | | | X | | | 5

| | | | | | X X | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | X X | | | | | | |

6 | | X | | | | | | | 6

| | X X | | | | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | | | | | |

7 | | | | | | | | | 7

| | | | | | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

0 1 2 3 4 5 6 7

[[[[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *0*], [[], *-1*], [[], *-1*], [[], *-1*]],

[[[], *-1*], [[*1*], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*]],

[[[], *-1*], [[], *-1*], [[*1*], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *0*], [[], *-1*]],

[[[], *-1*], [[], *-1*], [[], *-1*], [[*1*], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*]],

[[[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[*1*], *-1*], [[], *-1*], [[], *-1*], [[], *0*]],

[[[], *-1*], [[], *-1*], [[*2*], *-1*], [[*2*], *-1*], [[*2*, *1*], *-1*], [[*2*], *-1*], [[*2*], *2*], [[], *-1*]],

[[[], *2*], [[], *0*], [[], *1*], [[], *1*], [[], *0*], [[], *2*], [[], *1*], [[], *-1*]],

[[[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*]]]

**Estado inicial do jogo:**

Em Prolog, a representação das situações referidas até aqui é feita, como já foi referido, com base em listas e predicados especiais usados para mostrar um tabuleiro. Por exemplo, o tabuleiro do estado inicial do jogo acima seria:

[[[[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*]],

[[[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*]],

[[[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*]],

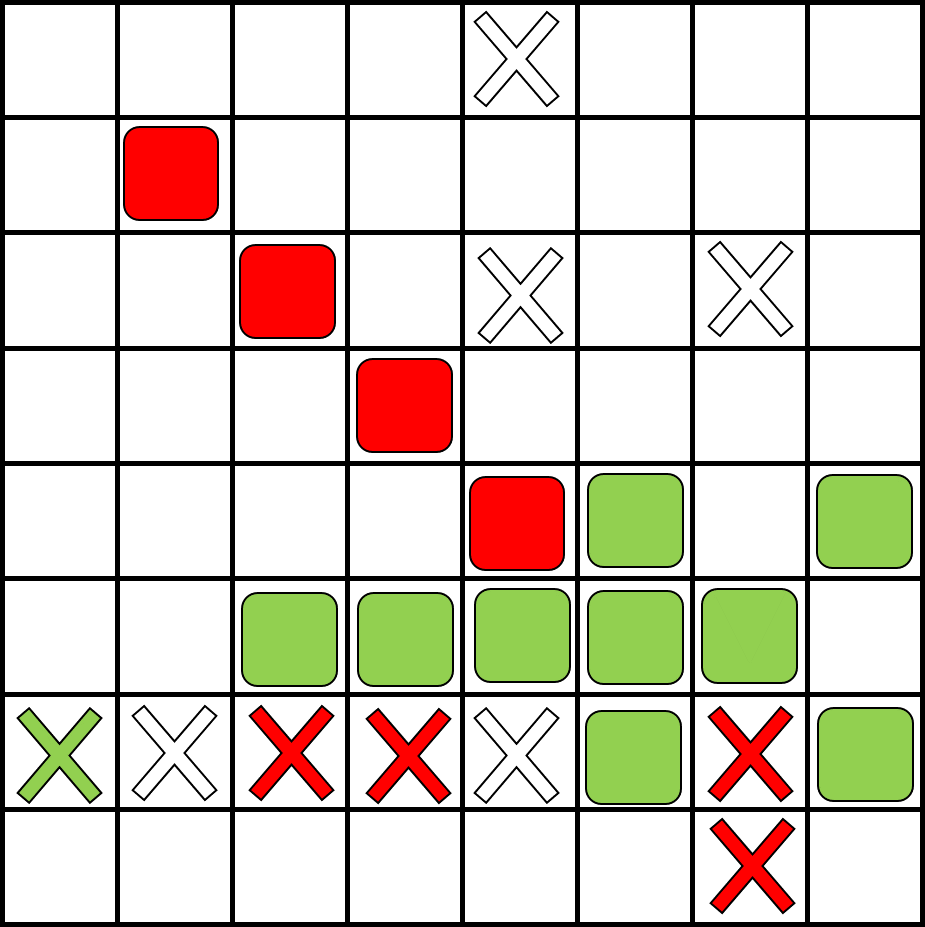
[[[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*]],

[[[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*]],

[[[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*]],

[[[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*]],

[[[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*], [[], *-1*]]]



0 1 2 3 4 5 6 7

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | | X X | | | |

0 | | | | | X | | | | 0

| | | | | X X | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | ##### | | | | | | |

1 | | ##### | | | | | | | 1

| | ##### | | | | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | ##### | | X X | | X X | |

2 | | | ##### | | X | | X | | 2

| | | ##### | | X X | | X X | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | ##### | | | | |

3 | | | | ##### | | | | | 3

| | | | ##### | | | | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | | ##### | ===== | | ===== |

4 | | | | | ##### | ===== | | ===== | 4

| | | | | ##### | ===== | | ===== |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | ===== | ===== | ===== | ===== | ===== | |

5 | | | ===== | ===== | ===== | ===== | ===== | | 5

| | | ===== | ===== | ===== | ===== | ===== | |

+-------+-------+-------+-------+-------+-------+-------+-------+

| » » | X X | \* \* | \* \* | X X | ===== | \* \* | ===== |

6 | » | X | \* | \* | X | ===== | \* | ===== | 6

| » » | X X | \* \* | \* \* | X X | ===== | \* \* | ===== |

+-------+-------+-------+-------+-------+-------+-------+-------+

| | | | | | | \* \* | |

7 | | | | | | | \* | | 7

| | | | | | | \* \* | |

+-------+-------+-------+-------+-------+-------+-------+-------+

0 1 2 3 4 5 6 7

**Estado final do jogo:**

O predicado utilizado para visualizar um tabuleiro é o “print\_board(+Board)”, no qual a variável “Board” representa uma lista semelhante à que se encontra acima, podendo diferir os valores numéricos no seu interior. O predicado e respetivos predicados auxiliares são visíveis no ficheiro “visualization.pl”.

Para além destes predicados, foi também implementado o predicado que inicializa o tabuleiro do jogo - “create\_board(-Board)” - que verifica se a variável “Board” contém um tabuleiro vazio (sem peças). Assim, pode ser usado para atribuir a uma variável esse tabuleiro. Este predicado encontra-se no ficheiro “board.pl”.

## 3.3 – Lista de jogadas válidas

É possível obter, para um determinado jogo, uma lista de jogadas possíveis através do predicado “available\_moves(+Game, -Moves)”. Este predicado assegura que, para um dado jogo “Game”, a lista “Moves” contém o conjunto de todas as jogadas que são possíveis de realizar. Neste caso, entende-se por jogada uma lista de 2 elementos que correspondem às coordenadas em X e Y da jogada disponível.

A implementação do predicado referido pode ser encontrada no ficheiro “plays.pl”.

## 3.4 – Execução de jogadas

A execução de uma jogada é feita com recurso ao predicado”place\_xpiece(+Game, +Coords, -New\_game) “. Este assegura que, dado um jogo “Game” e um par de coordenadas “Coords” (uma lista de dois elementos cujo primeiro corresponde à coordenada em X e o segundo à coordenada em Y), a variável New\_game contém um novo jogo resultante de se colocar uma peça X do jogador atual (esta informação é indicada na lista Game) nessas coordenadas. Este predicado garante que a célula onde se pretende colocar a peça está vazia, apenas sucedendo se isto for verdade.

A implementação do predicado referido pode ser encontrada no ficheiro “plays.pl”.

## 3.5 – Avaliação do tabuleiro

A avaliação do tabuleiro consiste em diferentes categorias. Para determinar as pontuações de cada jogador, é possível utilizar o predicado “game\_player\_scores(+Game, -Scores)” que garante que, para um dado jogo “Game”, a variável “Scores” contém uma lista de 2 elementos correspondentes, respetivamente, às pontuações do jogador 1 e jogador 2.

Para além deste predicado, existe também o predicado “game\_max\_score(+Game, -Score)” que garante que a variável “Score” contém a pontuação máxima para um jogador obter no jogo.

Existe também um predicado para calcular o valor de um jogo “Game” – “game\_value(+Game, -Value)”. Este predicado tem em conta o quão próximo o jogador atual está da pontuação máxima tendo também em conta a pontuação do adversário.

A implementação dos predicados referidos encontra-se no ficheiro “game.pl”.

## 3.6 – Final do jogo

Para verificar o se foi atingido o final do jogo existem dois predicados. O primeiro predicado é o predicado “game\_ended(+Game)” (ficheiro “game.pl”). Este predicado sucede se o jogo terminar ou porque algum jogador atingiu a pontuação limite (definida por “max\_score”, ver secção 3.6”) ou porque um dos jogadores ficou sem peças disponíveis. Existe também o predicado “game\_winner(+Game, -Winner)” (ficheiro “game.pl”) que determina o vencedor “Winner” de um dado jogo “Game” (“Winner” representa o valor 1 ou 2).

## 3.7 – Jogada do computador

Para avaliar a qualidade ou valor de uma lista de jogadas disponíveis e escolher, destas, a melhor, existe o predicado “ai\_evaluate\_and\_choose(+Moves, +MovePred, +ValuePred, +Game, -BestMove)”. A variável “Moves” deve conter uma lista com todas as jogadas possíveis (ver secção 3.3), “MovePred” deve ser o predicado que aplica a jogada ao jogo (ver secção 3.4), “ValuePred” deve ser a função que avalia o valor de um jogo (ver parágrafo anterior), “Game” deverá ser o jogo em questão. Dados todos estes valores, este predicado coloca em “BestMove” a melhor jogada a realizar. Para isto, o predicado utiliza predicados auxiliares (por exemplo, “ai\_evaluate\_and\_choose\_aux”) que avaliam o valor de todas as jogadas para determinar qual a que tem maior valor.

Existem dois níveis de dificuldade: um aleatório e outro que usa a função descrita acima para calcular o movimento que maximiza o “valor” do tabuleiro na jogada seguinte.

A implementação destes predicados encontra-se no ficheiro “ai.pl”.

# 4 – Interface com o utilizador

A interface com o utilizador utiliza predicados embutidos no SICStus Prolog para facilitar a introdução de informações pelo utilizador.

Para iniciar o jogo, apenas é necessário utilizar o predicado “play”, sem parâmetros (“play.”). Este mostra o menu inicial do jogo onde é possível escolher o modo de jogo : *1 vs 1*, *1 vs Bot* ou *Bot vs Bot*. Nos dois últimos casos, é pedido também ao utilizador para escolher a dificuldade do *bot*.

Para colocar uma peça no tabuleiro, seja ela um *joker* ou uma peça X normal, é pedido ao utilizador para inserir as coordenadas da posição de destino.

# 5 - Conclusões

Os objetivos iniciais foram cumpridos e foi possível terminar o desenvolvimento do jogo nos prazos estabelecidos.

Um dos problemas com que o grupo se deparou foi o facto de termos começado relativamente cedo, antes de aprender técnicas mais avançadas do Prolog, como o uso do *cut* (!) e de técnicas relativas a lógica de 2ª ordem. Isto obrigou a uma revisão do código, após a sua conclusão.

No sentido de melhorar o *bot* em modo difícil, uma vez que o algoritmo *minimax* é pouco eficiente e não aplicável, poder-se-ía alterar o predicado de avaliação do estado de jogo. Este poderia considerar, por exemplo, a existência de padrões parcialmente completos para tentar formar um padrão completo nesse local, ou para tentar bloquear essa possibilidade ao adversário.

# 6 – Bibliografia

* Página oficial do Mod X (<https://www.cryptozoic.com/games/mod-x>) (consultada a 7/11/2015 pelas 20:00)