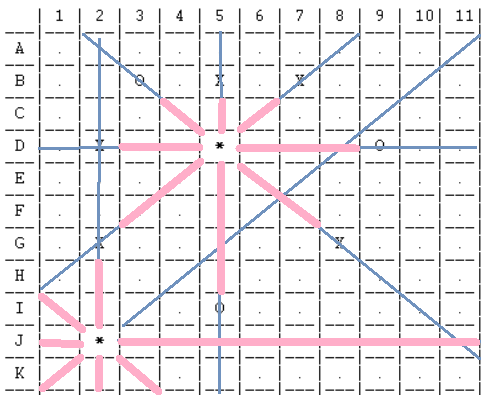
*Introdução*

Este projeto foi desenvolvido no Sistema de Desenvolvimento SICStus Prolog no âmbito da unidade curricular de Programação Lógica de 3º ano do curso Mestrado Integrado em Engenharia Informática e de Computação e tem como tema o jogo de tabuleiro Fabrik. O objetivo deste trabalho foi implementar, em linguagem Prolog, um jogo de tabuleiro e de peças, pelas regras de movimentação das peças (jogadas possíveis) e pelas condições de terminação do jogo com derrota, vitória ou empate. Foram implementados três modos de utilização: Humano/Humano, Humano/Computador e Computador/Computador.

Este relatório tem a seguinte estrutura:

* **O Jogo XXX:** Descrição do jogo e das suas regras.
* **Lógica do Jogo:** Descrição da implementação da lógica do jogo, tendo a seguinte estrutura:
  + **Representação do Estado do Jogo:** Exemplificação de estados iniciais, intermédios e finais do jogo.
  + **Visualização do Tabuleiro:** Descrição do predicado de visualização.
  + **Lista de Jogadas Válidas**: Descrição dos predicados usadas para a validação das jogadas.
  + **Execução de Jogadas**:
  + **Avaliação do Tabuleiro:** Descrição dos predicados que retornam o conteúdo do tabuleiro.
  + **Final do Jogo:** Descrição dos predicados que verificam o fim de jogo.
  + **Jogada do Computador:** Descrição dos predicados de geração de movimentos do computador.
* **Interface com o Utilizador:** Descrição do módulo de interface com o utilizador.

*Lista de Jogadas Válidas*

Uma posição é considerada válida para os jogadores (X/O) se a célula estiver vazia e se estiver na linha de visão de pelo menos um worker, isto é, a célula tem que estar num das 8 sentidos (N, S, E, O, NE, NO, SE, SO) em relação ao worker e entre essa célula e o worker não existir nenhuma peça. Na imagem à direita podemos ver um exemplo de tabuleiro onde as linhas azuis são as diferentes linhas provenientes do worker e a rosa as linhas de visão destes. Uma posição é considerada válida para o movimento dos workers se a célula se encontrar vazia.

Para a validação das jogadas foram usados os seguintes predicados que se encontram no ficheiro *logic.pl*:

checkMove(Board, Player, NewBoard, Expected, ColumnIndex, RowIndex)

isEmptyCell(Board, Row, Column, Res)

isValidPosLines(Board, Row, Column, Res)

isWorkerLines(Board, WorkerRow, WorkerColumn, Row, Column, Res)

verifyLine(Board, WorkerRow, WorkerColumn, Row, Column, 1, ResN, 'N' )

verifyLine(Board, WorkerRow, WorkerColumn, Row, Column, 1, ResNE, 'NE' )

verifyLine(Board, WorkerRow, WorkerColumn, Row, Column, 1, ResE, 'E' )

verifyLine(Board, WorkerRow, WorkerColumn, Row, Column, 1, ResSE, 'SE' )

verifyLine(Board, WorkerRow, WorkerColumn, Row, Column, 1, ResS, 'S' )

verifyLine(Board, WorkerRow, WorkerColumn, Row, Column, 1, ResSO, 'SO' )

verifyLine(Board, WorkerRow, WorkerColumn, Row, Column, 1, ResO, 'O' )

verifyLine(Board, WorkerRow, WorkerColumn, Row, Column, 1, ResNO, 'NO' )

**checkMove** – É chamado para verificar todas as jogadas (tanto do jogador, como do worker) recorrendo ao predicado *isValidPosLines* e *getValueFromMatrix.* Este ultimo serve para verificar se naquela célula do tabuleiro está o conteúdo pretendido (*Expected*). Por exemplo, se for para adicionar uma peça, o conteúdo pretendido é *‘empty’*, mas caso seja para escolher o worker a mover, o conteúdo pretendido vai ser ‘*red’*. Se não for possível fazer o movimento, este predicado chama o *write* que informa o jogador qual a razão da falha, pedindo umas novas coordenadas.

**isEmptyCell** – Verifica se a célula (*Row, Column)* está vazia recorrendo à chamadas ao predicado *getValueFromMatrix*.

**isValidPosLines** – Vai buscar as posições dos workers com a ajuda do predicado *getWorkersPos* e verifica se a célula (*Row, Column*) está na linha de visão de pelo menos um dos workers.

**isWorkerLines** – Verifica se a célula (*Row, Column*) está nalguma das linhas de visão do worker (*WorkerRow, WorkerColumn*) com a ajuda do predicado *verifyLine*.

**verifyLine** – Verifica se a célula está na linha de visão do worker. Foi feito o overload deste predicado para cada um dos sentidos com o intuito do código se tornar mais percetível.

*Avaliação do Tabuleiro*

Os predicados mais relevantes de avaliação do tabuleiro estão no ficheiro *utilities.pl* e são eles:

getWorkersPos(Board, WorkerRow1, WorkerColumn1, WorkerRow2, WorkerColumn2)

getValueFromMatrix([\_H|T], Row, Column, Value)

checkFullBoard(Board)

**getWorkerPos** – Percorre o tabuleiro e com a ajuda do predicado *getValueFromMatrix*, devolve em *WorkerRow1*, *WorkerColumn1*, *WorkerRow2* e *WorkerColumn2* as posições dos workers na matriz.

**getValueFromMatrix** – Analisa o que está na célula (*Row, Column*) da matriz. Retorna em *Value* o conteúdo daquela célula, ou, caso *Value* já esteja atribuído, a função falha.

**checkFullBoard** – Verifica se o tabuleiro está cheio, confirmando se não há nenhuma célula ‘*empty*’ no tabuleiro.

*Jogada do Computador*

Os predicados *startGame* e *gameLoop* recebem dois átomos *Player1* e *Player2*, que vão ser passados aos predicados *blackPlayerTurn* e *whitePlayerTurn*. Estes átomos podem ser *‘P’* (player) ou *‘C’* (computer). Assim é possível saber no ficheiro logic.pl, se se deve chamar os predicados relacionados com o computador.

Para a geração de jogadas do computador foram usados os seguintes predicados que se encontram no ficheiro *bot.pl*:

generatePlayerMove(Board, Row, Column)

moveWorker(Board, WorkerRow, WorkerColumn, WorkerNewRow, WorkerNewColumn, Res)

chooseWorker(Board, WorkerRow, WorkerColumn)

generateWorkerMove(Board,WorkerNewRow, WorkerNewColumn)

**generatePlayerMove** – Gera um linha e coluna aleatória, verifica se é uma jogada válida, ou seja, se é uma célula que está nas linhas de visão de algum worker e se a célula está atualmente vazia. Caso seja válida, ele devolve essa linha e coluna, caso contrário este predicado chama-se a si própria para tentar gerar uma nova posição.

**moveWorkerPos** – Escolhe aleatoriamente se vai mover ou não o worker. Se escolher mover o worker, chama o predicado *chooseWorker* para escolher aleatoriamente o worker a mover e seguidamente chama a função *generateWorkerMove* para saber a posição para qual mover*.*

**chooseWorker** – Escolhe aleatoriamente o worker a mover.

**generateWorkerMove** – Tal como o *generatePlayerMove*, gera uma linha e uma coluna e verifica se é valida. Neste caso é só verificar se a célula está vazia. Caso seja válida, ele devolve essa linha e coluna, caso contrário este predicado chama-se a si própria para tentar gerar uma nova posição.