Members:

CSINTSY S15

DESEMBRANA, Anna Patricia

MANDADERO, Clarissa Mae

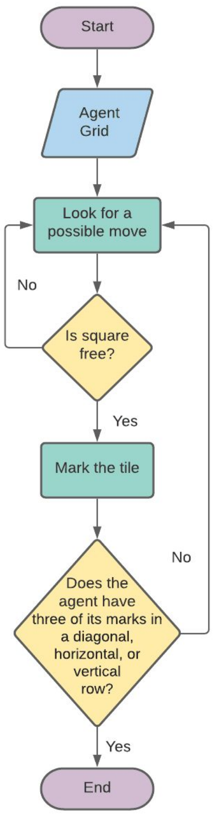
Technical Report on MCO2: Tic-Tac-Toe

1. **Define the goal of the TIC-TAC-TOE agent**
   - The goal of the TIC-TAC-TOE agent is to have three of its marks in a diagonal, horizontal, or vertical row.
2. **Formulate the problem**
   - TIC-TAC-TOE is played in a 3x3 grid. The two players marked as 'X' and 'O' alternately take turns in marking available tiles in the grid until one player has three of its marks in a diagonal, horizontal, or vertical row. The TIC-TAC-TOE agent has 3 levels of rational behavior: level 0, level 1, and level 2. In level 0, the agent makes random, but valid moves. In level 1, the agent uses a hard-coded table that generates a move for every possible configuration. While in level 2, the agent uses heuristics to find the best move given the current configuration.
3. **Determine the specific states and configurations that the agent operates on.**
   a. States:
      - A 3x3 grid wherein each tile could either be free, marked by 'X', or marked by 'O'.
      - For the first playthrough, the player goes first. Every time the player decides to play again, the first move will be alternating between the player and the computer.

   b. Initial State:
      - The game starts with a 3x3 grid wherein all tiles are marked as free.

   c. Actions:
      - Mark a tile

d. Transition Model:


Start
Agent Grid
Look for a possible move
Is square free?
No
Yes
Mark the tile
Does the agent have three of its marks in a diagonal, horizontal, or vertical row?
No
Yes
End

e. Goal Test:

- If a player has three of its marks in a diagonal, horizontal, or vertical row.

f. Path Cost:

- The path cost is the total number of possible winning lines.
- In the perspective of the computer, when the player has the advantage, the path cost is -1. When the computer has the advantage, the path cost is 1. When it is a tie or when there are no available spots left in the board, the path cost is 0.

4. **Determine the specific actions, and the states (configurations) on which the actions are applicable (include illustrations for transition table)**

Name	Condition	Transition	Effect
Mark tile	board[row][col] = ‘ ‘	board[row][col] = ‘X’	emptySpots--
	board[row][col] = ‘ ‘	board[row][col] = ‘O’	emptySpots--

5. **Specify how the goal state can be determined and detected by the agent**

This TIC-TAC-TOE agent has 3 levels of rational behavior: level 0, level 1, and level 2. In level 0, the goal state can be determined and detected through the use of random actions. It does not make use of any logical algorithm, nor any heuristic approach.

For level 1, the goal state can be determined and detected through the use of a hard coded table which specifies the move that the agent could do given the current board configuration.

In level 2, the goal state can be determined and detected using the Minimax Algorithm. Using this algorithm, the best move for the agent is found by recursively finding the terminal state with the best outcome.

6. Describe the three levels of rational behavior

In level 0, the agent makes random, but valid moves. It does not take into consideration the current board configuration. It also does not use any logical algorithm, nor any heuristic approach.

In level 1, the agent bases its moves on a hard coded table wherein it checks which possible move could be the best move given the current board configuration. It does not take into consideration future moves, but only focuses on the current state of the board. For the first move of the agent, it would take the middle tile (1,1) since the middle tile has the most number of winning combinations. If it has been marked by the opponent, then the agent would take the upper right tile (0, 2) since it has the second most number of winning combinations, together with the other 3 corner tiles. All other moves of the agent will now be based on the hard-coded table that we have created. To simplify, the agent will look for the best possible move given the current board configuration, and to do this, we have created 4 conditions. The first condition checks if there is a free tile that upon being marked by the agent would result in the agent's win. If there is such a tile, then it would mark that tile as its own. The second condition checks if there is a free tile that once ignored could result in the opponent winning. If there is such a tile, then the agent would mark it as its own. The third condition checks if there are two adjacent free tiles whose third adjacent tile is taken by the agent. If there is such a tile, then the agent would mark it as its own. The fourth condition checks if there are two adjacent free tiles whose third adjacent tile is taken by the opponent. If there is such a tile, then the agent would mark it as its own. If ever none of the conditions is applicable to the current board configuration, then the agent will just choose any free tile.

In level 2, the agent determines its best move by using the Minimax Algorithm. First, the agent checks if there is a spot available in the game board. If there is an available spot, it temporarily places a move at that spot and uses the algorithm to get the score (whether that move gives an advantage to the computer or the player or results in a tie). Basically, what the algorithm does is that it recursively checks all the possible states until it reaches a terminal state. Once it reaches a terminal state, it will return the score(1 if the computer wins; -1 if the player wins; and 0 if it is a draw) as it backtracks. The possible state with the best score will be determined by comparing each possible state's score with one another. Once all the possible state's are checked, the best move remains and is chosen by the agent.