Department of Computer Science & Software Engineering

SOEN 6011 Summer 2016

State Machine Diagram for Heuristic in Tic-Tac-Toe Project

Assignment 5

Professor: - Nicolangelo Piccirilli

Team Name: Triple – T
Group Number - 1
Date of submission – 3rd June 2016

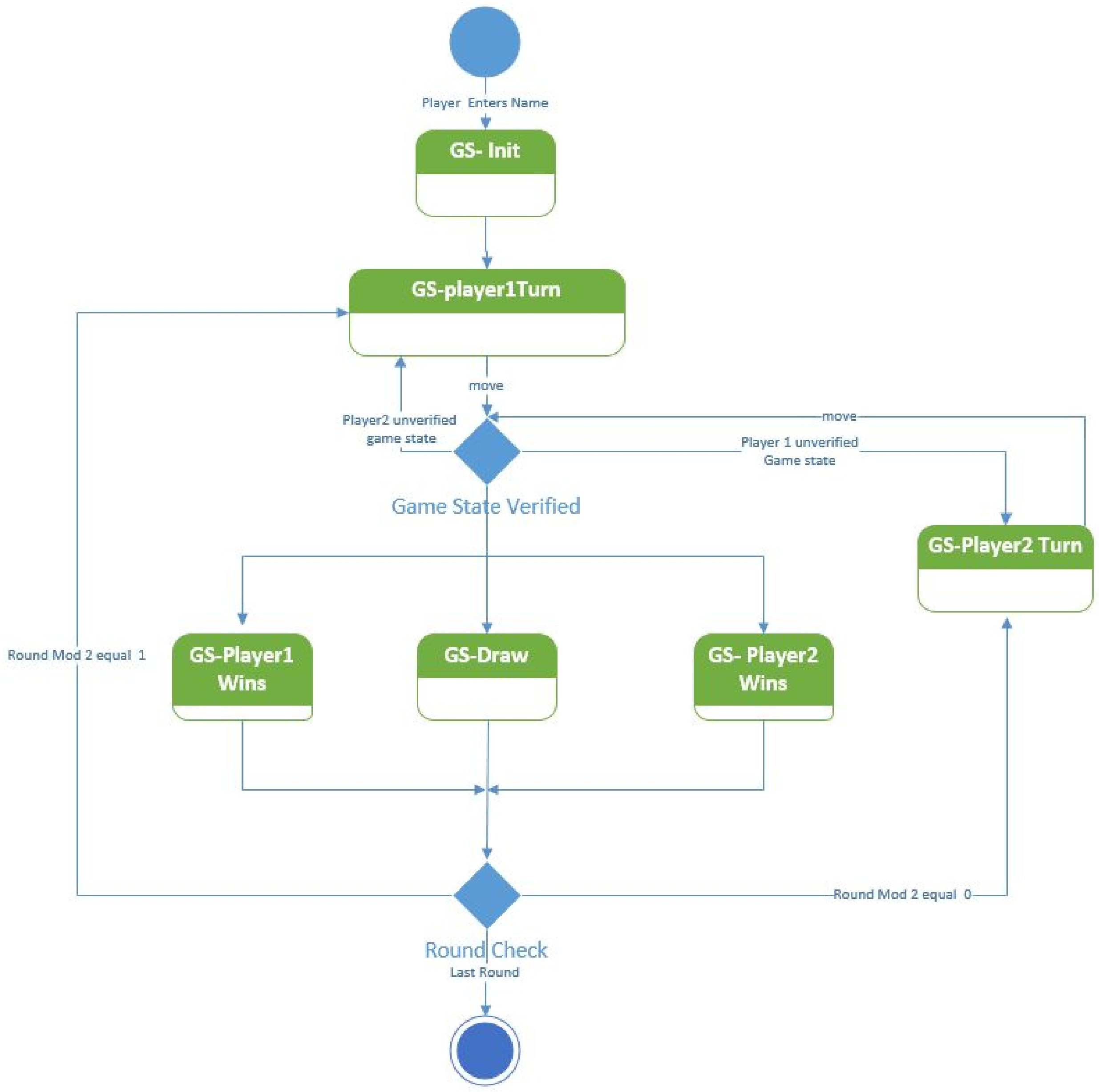
Team Leader: NIDHI ARORA

Team Members:

	Name	Student ID	Contribution
1.	Amanjot Kaur Ahluwalia	40011623	Requirements Analysis - Documentation
2.	Arash Arasteh	40000580	Coding Team 1- Documentation
3.	Arash Farkish	27678835	Coding Team 1- Documentation
4.	Basireddy Sandeep Kumar	40016071	Testing - Designing
5.	Dalvir Singh Bains	40012722	Requirements Analysis - Documentation
6.	Nidhi Arora	40014504	Coding Team 2 - Documentation
7.	Parinaz Barakhshan	27675518	Coding Team 1- Documentation
8.	Sarthak Batra	27408978	Coding Team 2 - Documentation

Table of Contents

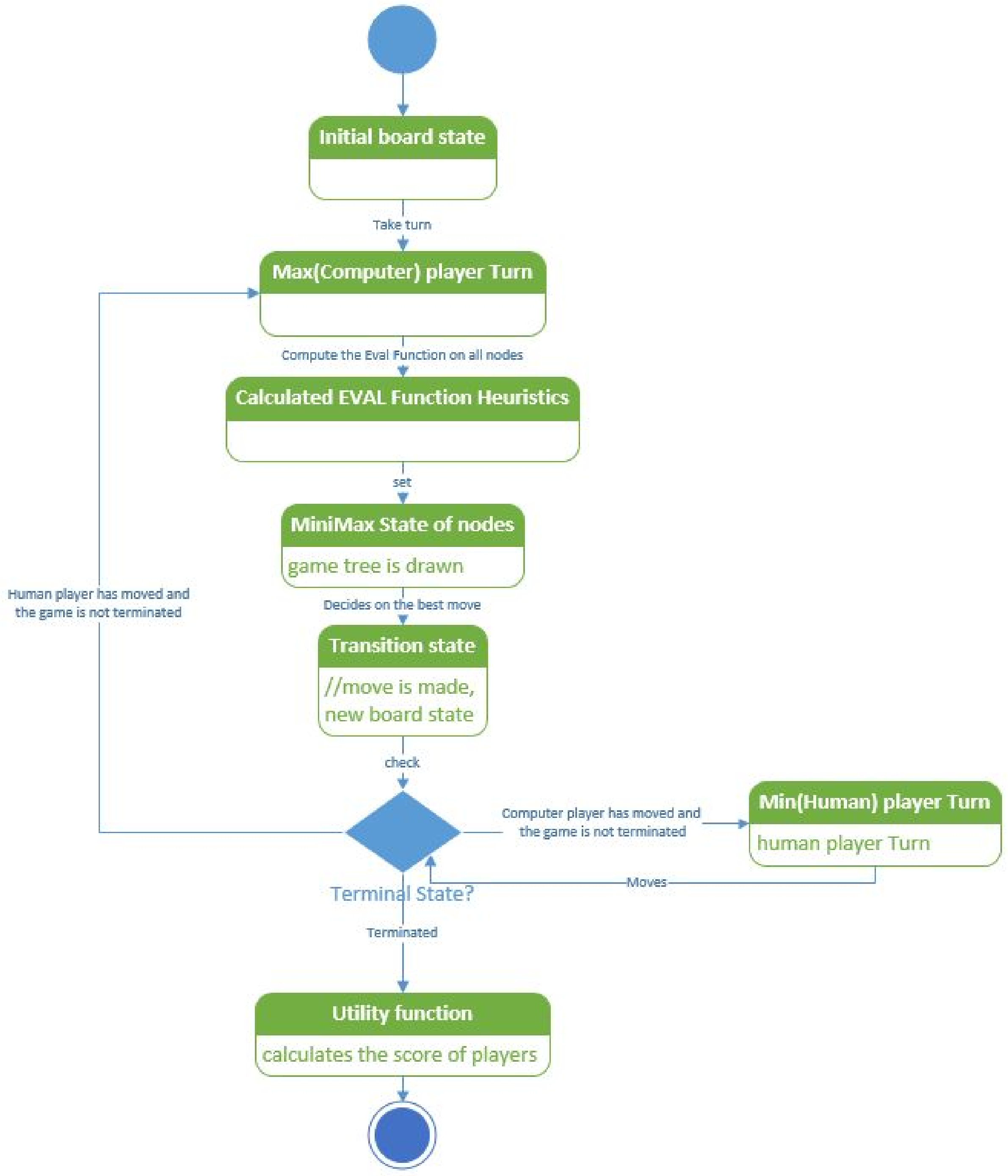
State Machine Diagram for Tic-Tac-Toe game



Here, GS stands for Game State.

Our objective is to model a state machine diagram that shows the different states the game heuristic goes through when computer player attempts to beat human player.

State Machine Diagram for game Heuristics


Initial board state
Take turn
Max(Computer) player Turn
Compute the Eval Function on all nodes
Calculated EVAL Function Heuristics
set
MiniMax State of nodes
game tree is drawn
Decides on the best move
Transition state
//move is made, new board state
check
Human player has moved and the game is not terminated
Computer player has moved and the game is not terminated
Min(Human) player Turn
human player Turn
Moves
Terminal State?
Terminated
Utility function
calculates the score of players

Assumptions:

- Two agents whose actions alternate.
- Utility values for each agent are the opposite of the other.
- Fully observable environments.
- “ply” = one action by one player, “move” = two plies.

State Machine Description for game Heuristics:

We have two players: MAX(Computer) and MIN(Human). The MAX moves first and the players take turn until the game is over. The winner gets reward, loser gets penalty. The states are as below:

1. **Initial Board State:** Set-up specified by the rules, initial board set-up.
2. **Max(Computer) Player Turn:** Defines that the computer player has the move in a state.
3. **Calculated Heuristic Evaluation Function:** Evaluation Function estimates how good the current board configuration is for a player. Evaluation function shows the estimate of cost from start to goal through given node.
4. **Minimax State of Nodes:** Sets the state of the each node on the Game Tree
5. **Transition State :** Defines the result of a move.A new board state.
6. **Terminal Test:** Is the game finished? True if finished, false otherwise.
7. **Min(Human) Player Turn:** Defines that the Human player has the move in a state.
8. **Utility Function:** Gives numerical value of terminal state for the player . win (+1), lose (-1), and draw (0).

Game Heuristic Model Description:

Here is the flow that we modeled our Game:

- **Game tree** represents alternate computer/opponent moves.
- **Static heuristic evaluation function**:Estimates quality of a given board configuration for the Max player.
- **Minimax** is a procedure which chooses moves by assuming that the opponent will always choose the move which is best for them. Basically, it avoids all worst-case outcomes for Max, to find the best.If the opponent makes an error, Minimax will take optimal advantage of that error and make the best possible play that exploits the error.

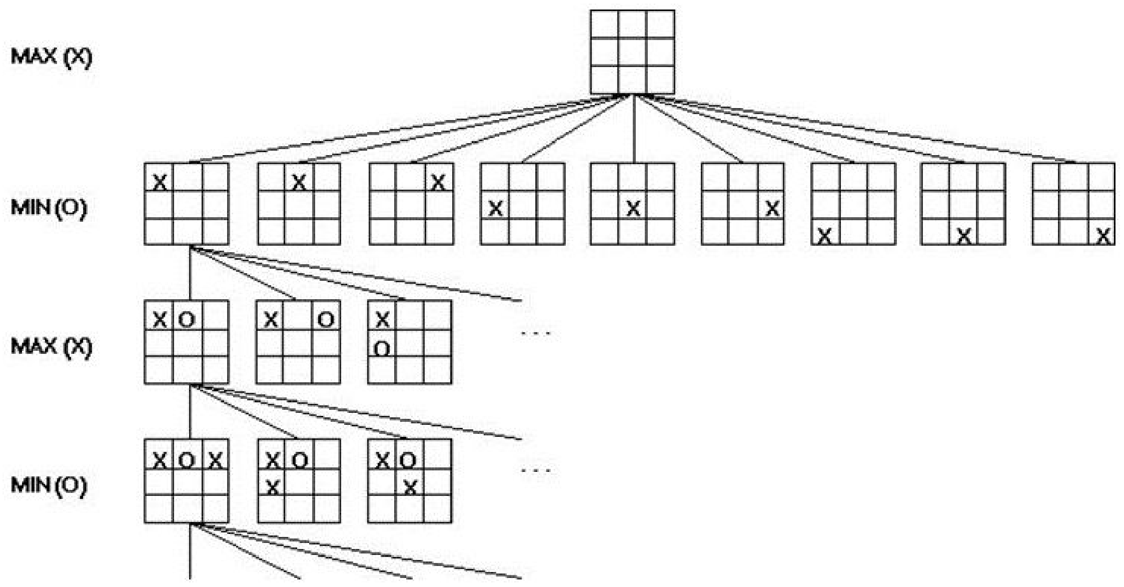
How to find the optimal strategy and best next move

We have two players: MAX and MIN. MAX moves first and they take turns until the game is over. MAX uses game tree to determine “best” next move. To find the optimal strategy and best next move for Max, we go through these steps:

1. Generate the whole game tree, down to the leaves.

A game tree is a directed graph whose nodes are positions in a game and whose edges are moves. The complete game tree for a game is the game tree starting at the initial position and containing all possible moves from each position.

Partial Game Tree for Tic-Tac-Toe



The diagram shows the first three levels, or plies, in the game tree for tic-tac-toe. The rotations and reflections of positions are equivalent, so the first player has three choices of move: in the center, at the edge, or in the corner.

Game Tree Size for Tic-Tac-Toe

Each player can have 4 or 5 legal actions per state on average, total of 9 plies in game.
5^9 = 1,953,125
9! = 362,880 (if Computer goes first)
8! = 40,320 (if Computer goes second)

(Static) Heuristic Evaluation Functions

An Evaluation Function estimates how good the current board configuration is for a player. Typically, evaluate how good it is for the player, how good it is for the opponent, then subtract the opponent's score from the player's. Often called "static" because it is called on a static board position.

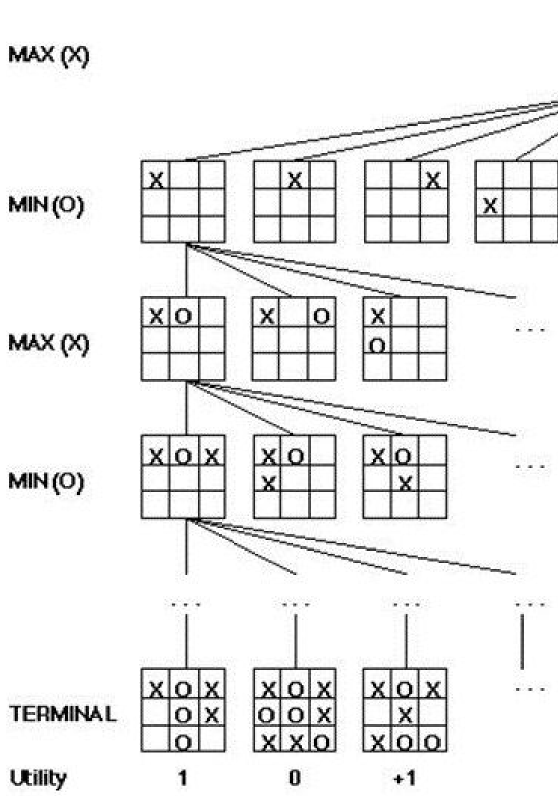
- Eval(n)=X(n)-O(n)

- X(n) is the total of X’s possible winning lines
- O(n) is the total of O’s possible winning lines
- Eval(n) is the total Evaluation for state n

Returns one value, positive for advantage to one player, Negative means advantage to the other. Zero indicates it is even.

2. Apply utility (payoff) function to each leaf.

Partial Game Tree for Tic-Tac-Toe



- f(n) = +1 if the position is a win for X.
- f(n) = -1 if the position is a win for O.
- f(n) = 0 if the position is a draw.

3. Back-up values from leaves through branch nodes:

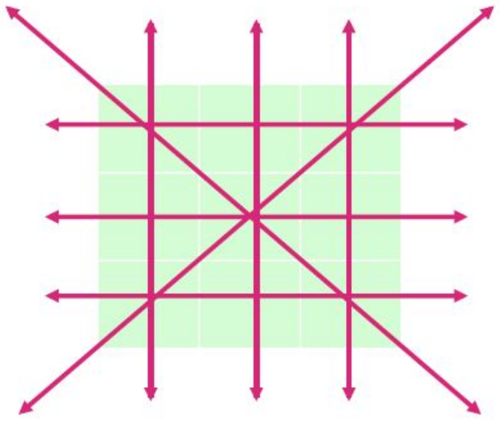
–a Max node computes the Max of its child values

–a Min node computes the Min of its child values

4. At root: Choose move leading to the child of highest value.

Sequence of Legal states

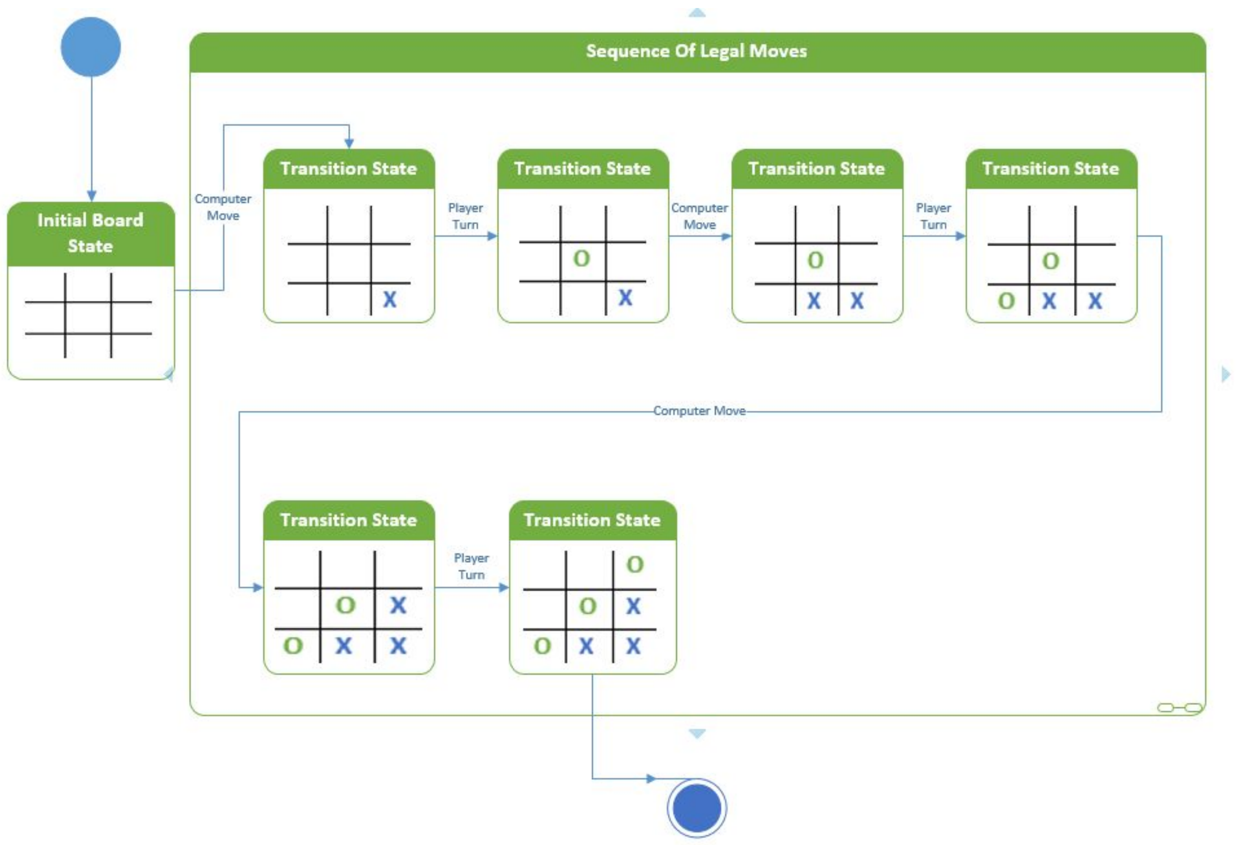
On an empty game board there are 8 possible winning lines for both X and O which can be vertical,horizontal or Diagonal.



Here we are going to show some samples for terminating state of the game,which can be Player1 Win, Player 2 Win, or Draw. For the node transition state in between MiniMax state of nodes and terminal test above in the State Machine Diagram for game Heuristics there are 9 possible ways of moves which ultimately gives us either a win or a draw between the computer and the person who play’s. We have 2 possible ways winning by diagonal, 3 vertical, 3 horizontal, finally a draw. We have shown in the below diagrams.

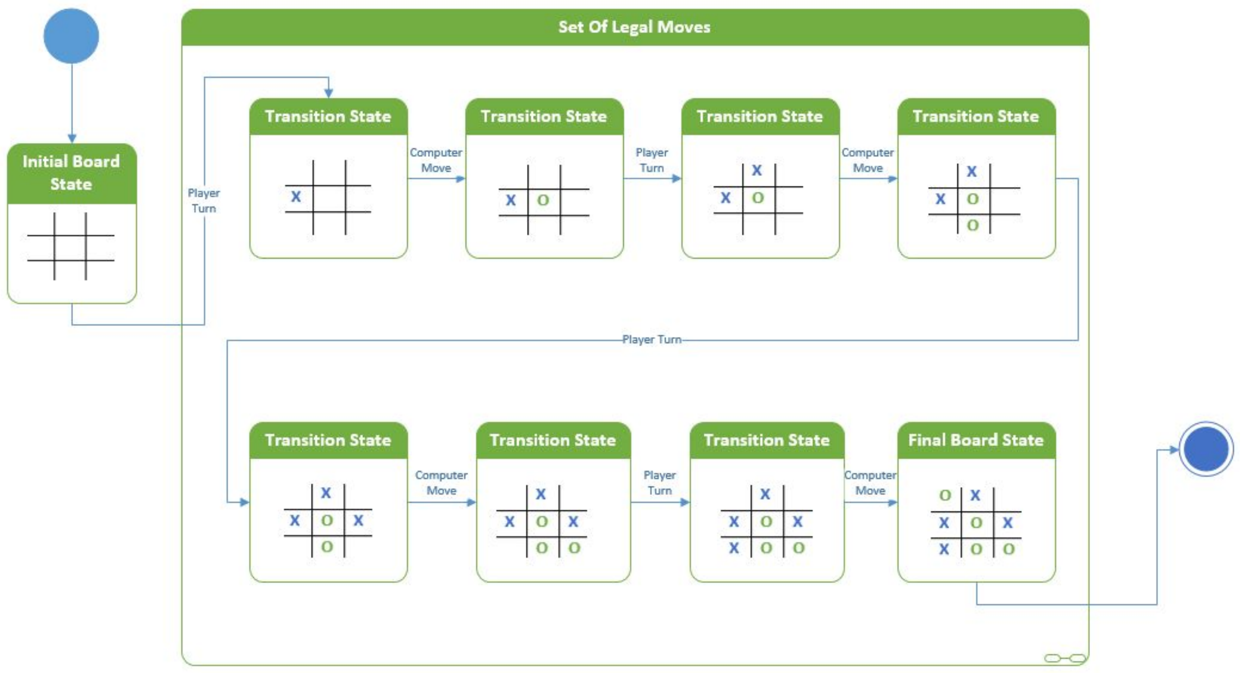
Transition States between MiniMax state of nodes and terminal test

Win Condition-(Diagonal 1)



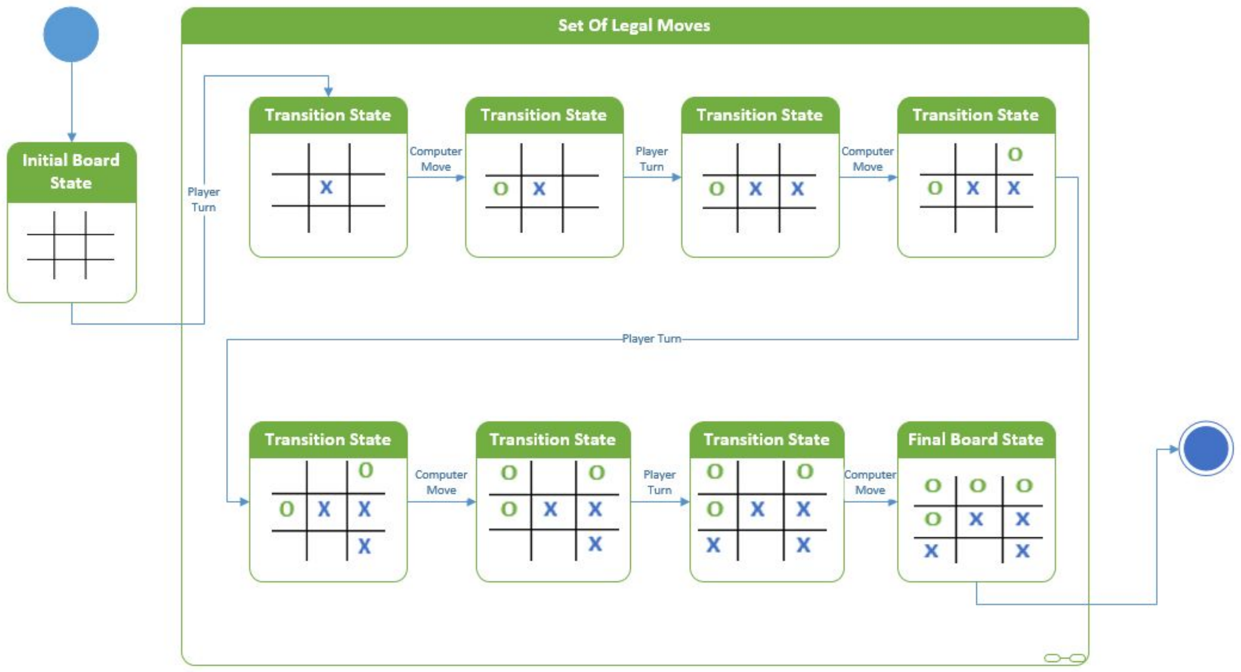
O represents Computer move, by using Heuristic Evaluation Function.
X for Human player move.

Win Condition-(Diagonal 2)



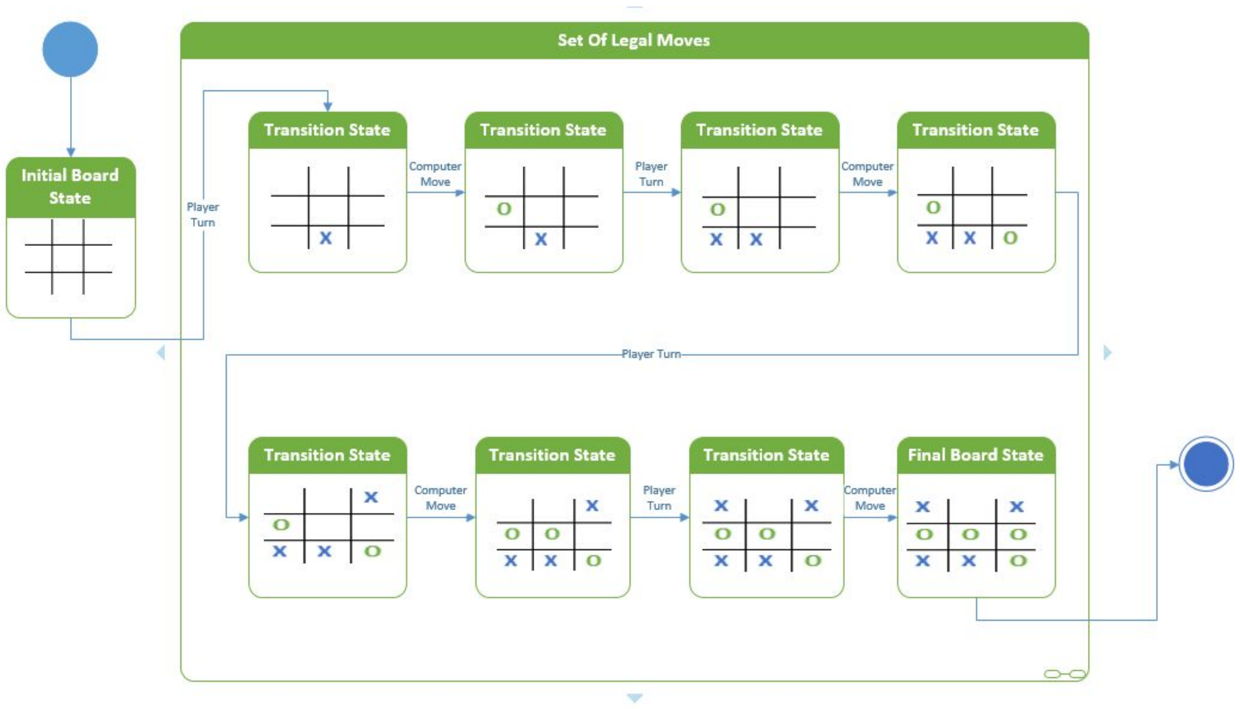
O represents Computer move, by using Heuristic Evaluation Function.
X for Human player move.

Win Condition-(Horizontal 1)



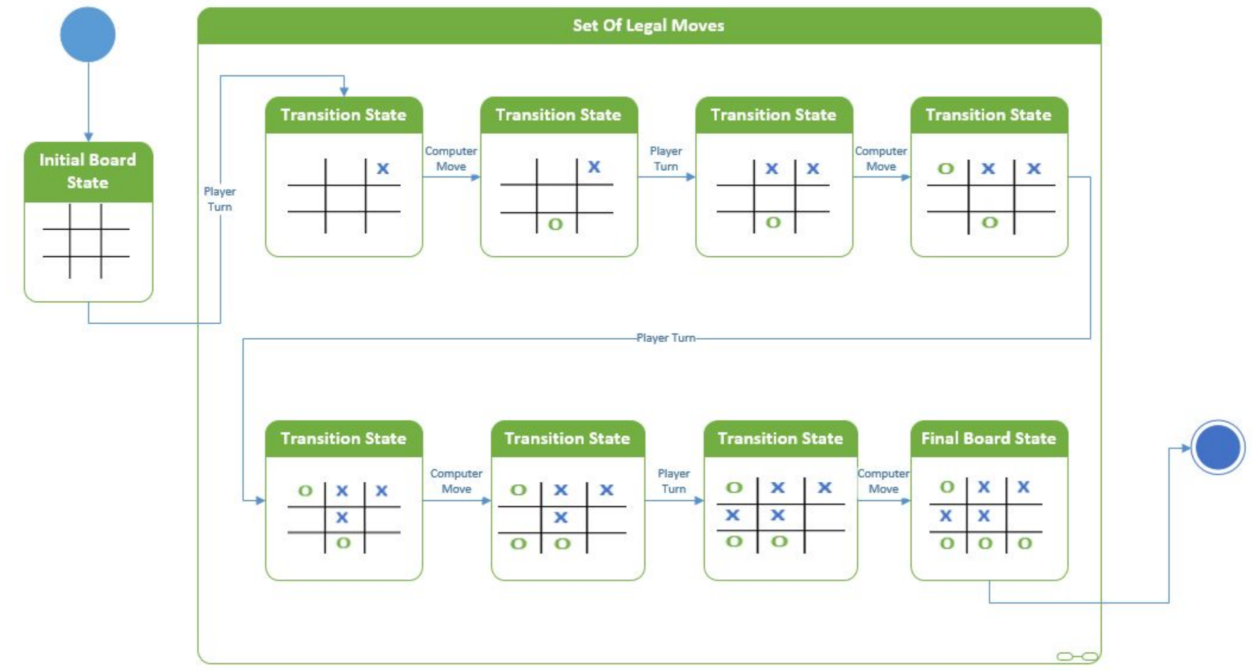
O represents Computer move, by using Heuristic Evaluation Function.
X for Human player move.

Win Condition-(Horizontal 2)



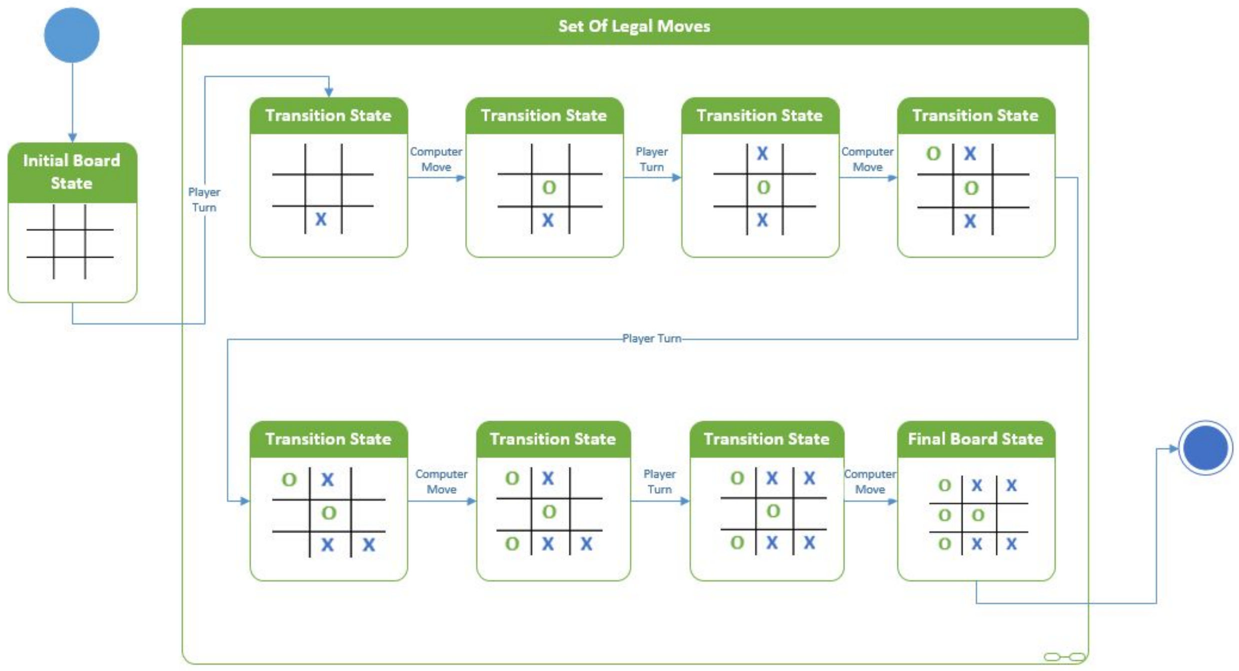
O represents Computer move, by using Heuristic Evaluation Function.
X for Human player move.

Win Condition-(Horizontal 3)



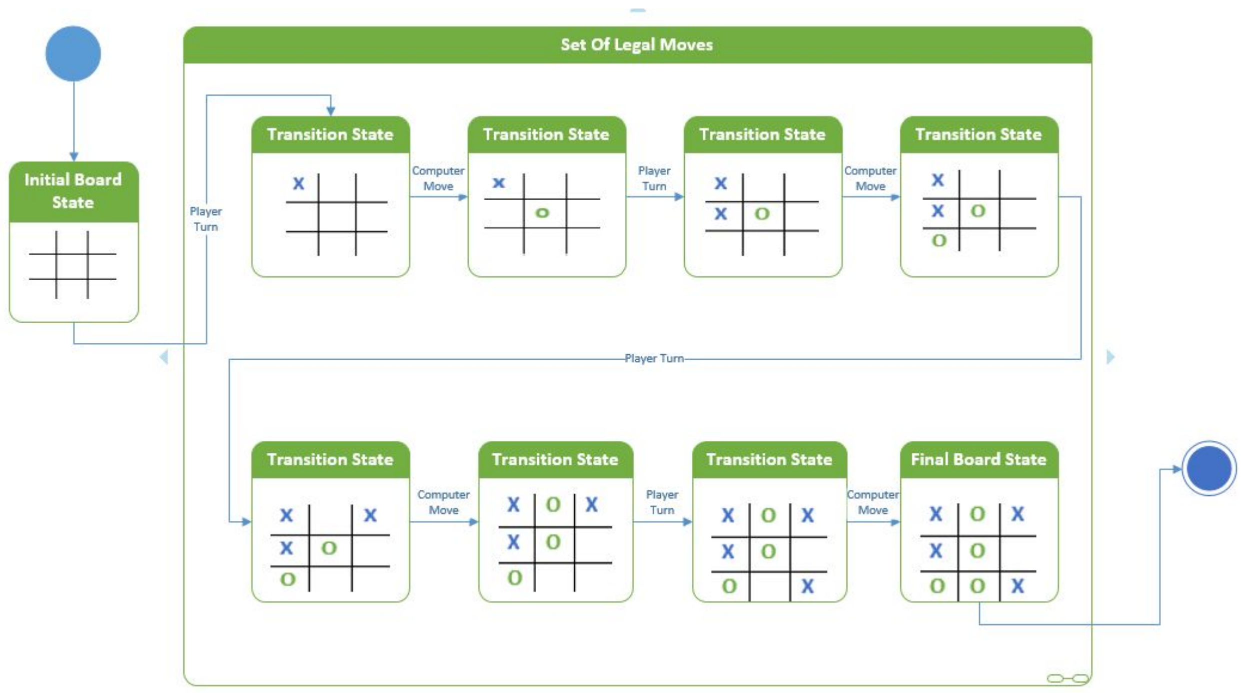
O represents Computer move, by using Heuristic Evaluation Function.
X for Human player move.

Win Condition-(Vertical 1)



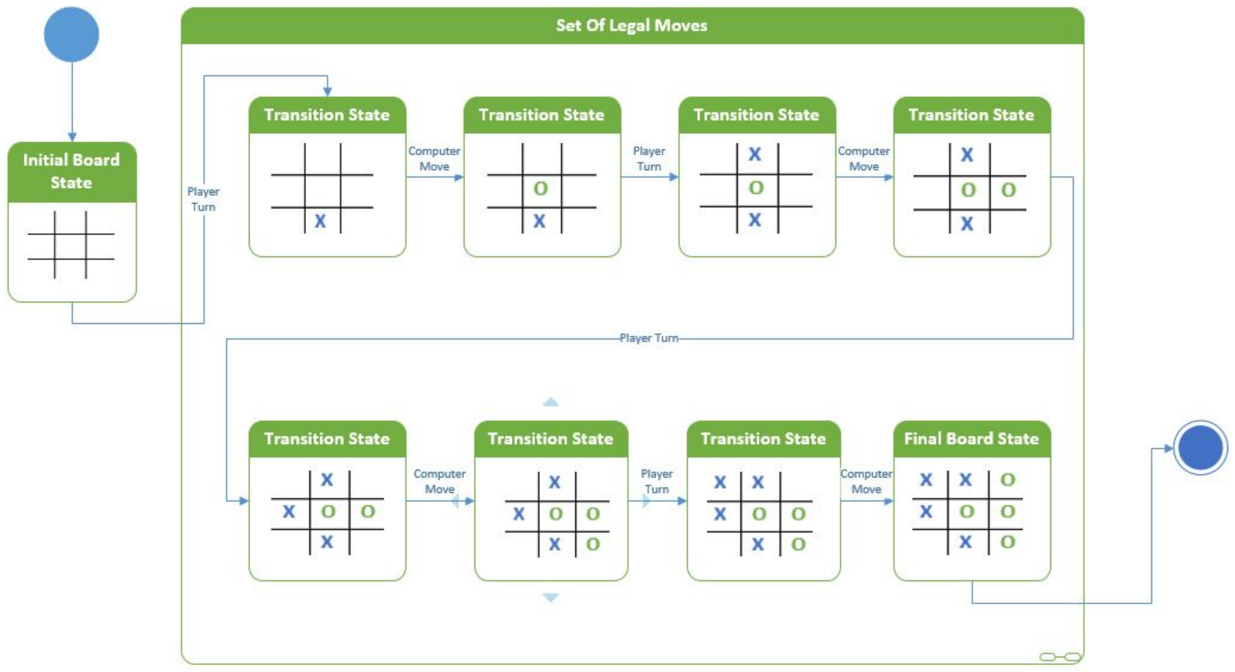
O represents Computer move, by using Heuristic Evaluation Function.
X for Human player move.

Win Condition-(Vertical 2)



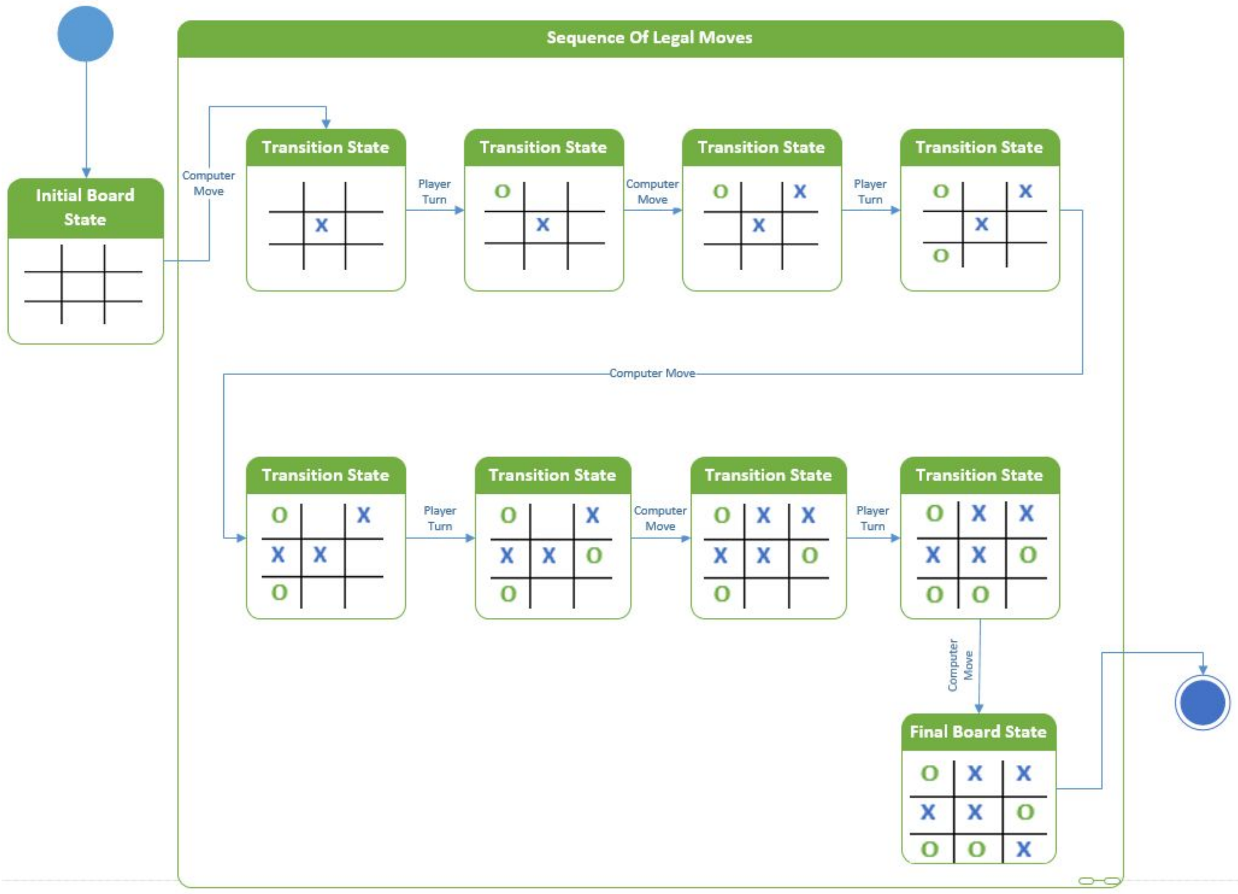
O represents Computer move, by using Heuristic Evaluation Function.
X for Human player move.

Win Condition-(Vertical 3)



O represents Computer move, by using Heuristic Evaluation Function.
X for Human player move.

Draw Condition



O represents Computer move, by using Heuristic Evaluation Function.
X for Human player move.