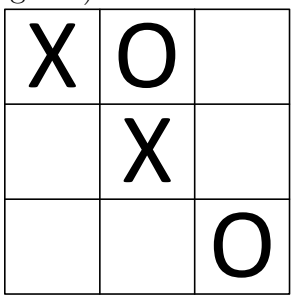
Homework 01 Solutions

ECE 587, Spring 2025

Note that there is no single "correct" solution to the homework questions. I would suggest you to compare your model with my model in order to understand if your model is reasonable or not.

In this homework, we will study the FSM model to build an interactive tic-tac-toe game. Since our focus will be the FSM model itself, we don't care how the game will be implemented, i.e. we don't care what language should be used and you are not required to actually implement the game.

The tic-tac-toe game, as shown below, uses a 3-by-3 board. Two players, X and O, take turns to fill an empty board until one of them wins by placing the same three in a row (horizontal, vertical, or diagonal).



As a hint, let's use the board itself as the state of our FSM model. Here are the questions you'll need to answer. The answers should be concise, say 3 to 5 sentences at most.

1. *(1 point)* How many states are there in the FSM model?
   The board has 9 boxes and there are three possibilities for each box (empty, X, and O). So there are 3^9 *states.*

2. *(1 point)* What are the initial states and the final states?
   There is a single initial state where each box is empty. There are many final states where one of the players wins, or the game ties if all boxes are filled with O or X.

3. *(1 point)* What is the output function like?
   The output function displays the board with X's and O's as indicated by the current state. For example, the output function could generate a binary string to drive a matrix display.

4. *(1 point)* What are the inputs? What is the next-state function like?
   Each input indicates what (X or O) should be played next and where it should be placed (among the 9 boxes). The next-state function may reject an invalid input by staying at the currect state, or move to the next state by adding the X or O to the box.

5. *(1 point)* We would like to have the ability to inspect any previous moves by the players in the current game. Describe how this could be achieved by the FSM model.
   Just use an array or a linked list to store the past inputs. Starting from the initial state, we may replay any number of steps.

Though not required, for those who would like to see the actual code, please refer to the page `https://reactjs.org/tutorial/tutorial.html` .
If you haven't done so, I would recommend you to spend sometime on this example to understand how state machines help to build graphical user interfaces.