Abstract:

Tic-tac-toe was one of the first solved games. It’s a simple game with less than 3^9 (each square can be x, o or blank) possible board states, which can be drastically reduced when the AI has control of one player. I have made a DFA that will play tic-tac-toe as the first player. The AI exhibits perfect play and will only draw a game 4 out 91 possible completed board states. While technically, the language only needs a series of O’s moves, however the accepted input will also contain X’s prior moves for better readability. The demo is interactive in that it will display the current board state both as the string and as the full board. It will then prompt the user for their move, and assuming its valid, will display the new board state with the AI’s move as well.

Introduction:

Tic-tac-toe is a simple children's game where people put x’s or o’s on a 3x3 grid. It’s one of the first solved games, and computer exhibit perfect play. While it may look like there are 3^9 possible board states, this number can be drastically reduced as the computer will always make the same move. The DFA only needed 118 different states, which includes an error state. Play starts with the AI moving in the top left corner and the opponent can move in any other open spots on the board. Each turn the current state is printed out which is the sequence of moves that have been made.

Detailed System Description:

Users moves are valid with ‘O[r][c]’ with r as the row and c as the column for the position they want to move. Technically the language only needs move position but also inputting the player improves readability. A sample of the DFA is posted below, starting with board state X00O22X02.

Requirements:

The delta transition table (implemented as an EnumMap in java) has a space requirement of O(states * acceptedInput) with states = 119 and acceptedInput = 9. The number of accepting states is 91.

User Manual:

Running Game.java will prompt the user to input a string interactively. This string will be only the piece of consumed input (not the full input string). Running Test.java will validate lines of full input and for each line respond valid or invalid (currently incomplete).

Conclusion:

It solves Tic-Tac-Toe.