# READ ME

I. Software Introduction

This software is called FSA Creator and FSA Solver. The FSA Creator will create a text file that will contain a string representation of the Finite State Automaton that you create through the GUI. The FSA Solver will allow you to load an FSA and solve it there or create a LISP Solver or PROLOG Solver of the loaded FSA. Once you create a LISP program through the FSA Solver, start up a LISP environment and load up the file you made to test it out! The same can be said for the PROLOG program. xLisp was used for testing the created LISP program and PLWIN was used to test the PROLOG program. The function/rule name for the FSA is (fsa x) for LISP and fsa(x) for PROLOG.

II. Installation

You will need the following files to run FSA Creator.

FSACreator.java  
InfoPanel.java  
FSADisplay.java  
FSASolver.java  
FSADraw.java  
State.java  
Transition.java  
LISPCreator.java  
PROLOGCreator.java

To run the program through command line, first put all of the files into a folder. Then go into that folder on the command line shell and type “javac \*.java”. You will now be able to run FSACreator or FSASolver. The difference between the two is that you can create an FSA on the fly and create an FSA file with FSACreator. When using FSASolver, you can load up an FSA file that you created through FSACreator and either solve it there or turn it into a LISP program that can solve the FSA by selecting “LISP”or turn it into a PROLOG program by selecting “PROLOG”.

III. Run Software

First, you will need to create an FSA that you can test against. To do so, be in the folder with the files on the command line shell. Type in “java FSACreator” and this will pop up an FSACreator window.

Input the correct information for your FSA as needed.

**Alphabet**: character, character, etc.  
Example: a,b,c,d

**Transitions**: put in the transition’s begin state, next state and next character and then press “Add”. Do this as many time as you need to create your transitions in the FSA.

When you’re done creating the FSA, press the “Translate” button to see the FSA. The FSA’s string representation will appear in the text box above the button for you to double check your work. If it’s correct, select “Yes” and you will be prompted to give your FSA a name.

To solve an FSA, type in “java FSASolver” and a GUI will pop up with buttons like “Load”, “LISP”, and “Solve”. First, you need to load an FSA onto the software. When done, an FSA will appear in the appropriate text box. You can now either solve the FSA here or turn it into a LISP program.

If you wish for your FSA to be runnable in a LISP environment, select the “LISP” button on the software and you will have to create a name for your LISP program. Be sure to add the “.lsp” extension to the end of the file name.

Example: “testMe.lsp”

Now you can run this program in a LISP environment and test a list of characters (or string) against the program. It will return a statement that the string was accepted if the string is accepted, or a statement that it was rejected if not. To try it out, use the function (fsa x).

If you wish for your FSA to be runnable in a PROLOG environment, select the “PROLOG” button on the software and type in a name for your PROLOG program. Remember that the extension needs to be “.pl”.

Example: “testMe.pl”

To try it out, use the functor named “fsa(x).”.

