Christopher Ray
Marina Blanton
CSE 30151
9 February 2015

Project 1 Report

I wrote this program in C++ and tested on both an Asus Zenbook (4 GB, Intel i5) running Ubuntu 14.04 (32-bit) and the CSE student machines.

The program works by first reading from an input file in order to fill in several vectors of the nfa object. These vectors contain the nfa's alphabet, set of all possible states, transitions, starting state, and final accept states. The transition vector contains a series of structures the hold information about the state to transition out of, the necessary input for a transition, and the state to transition to. The program then begins to read from standard input (which can be in the form of a text file fed in using the ' < ' operator) once it has finished reading from the input file. The first character read in from the standard input should be an integer equal to the number of tape inputs that are to be fed to the nfa. All other inputs after that initial integer are then treated as inputs to the nfa. The nfa will read in an entire tape input, tokenize it in order to get each individual input character, and finally iterate through the transition vector, modifying a member vector called currentStates every time a transition is made. The program will continue to iterate through the transition vector and update the currentStates vector until it is impossible to perform any more transitions from a given input character, at which point the contents of the currentStates vector will be printed to the standard output along with the corresponding character input. Once the end of a tape input has been reached, the program checks whether or not the a member of the currentState vector is a final accept state and will print “ACCEPT” or “REJECT” based on the result.

I first tested the program using the nfa1.txt and nfa.txt files, along with their corresponding tape input files, provided to students in order to confirm that the nfa was properly being built, reading in input, and making the correct transitions. Once I was able to get outputs similar to the sample output files provided to students, I then tested the program using test-case1.txt, which contains the nfa description for the language 0*{01}*, and test-case2.txt, which contains the nfa description for the language {ε, 0, 01+} and their respective tape input files, case1-input.txt and case2-input.txt. Those input files contain several strings from the described languages as well as several strings that should be rejected based on the described languages, all of which the program was able to correctly calculate.

The main difficulty I encountered with this project was getting the transition function for the nfa to properly work. The main issue involved expanding the currentStates vector in response to transitions caused by the empty string. Initial attempts to account for empty string inputs resulted in multiple incorrect ACCEPT's and REJECT's, states reappearing in or disappearing from the currentStates vector, and other random errors that somehow only appeared once. The current implementation of the transition function will first transition based on a character input before attempting to transition based on empty string input. In other words, the nfa will attempt move as far as it can through the states based on the current input.