Samantha Rack
CSE 30151 – Theory of Computing
Project 1

Language and Platform:

The language in which I developed this project is C++. The program was developed on one of the student machines, so the platform was Linux Redhat.

High Level Description:

The simulation program begins by creating an NFA object from the description file. After reading the first line of input from the standard input, the program enters a loop to first save each line of inputs to be processed, and then process each of the inputs symbol by symbol in the NFA. Between each processing of inputs, the machine is reset to its start states.

The major data structures in the NFA class hold each of the components of the machine. The alphabet, set of states, set of accept states, and current states are stored in arrays of strings. The most notable data structure is the transition function, which contains an array of the next states for each input, state pair.

On each input, the program copies each of the current states, clears the current states, and iterates through each of these states using the transition function data structure to determine what states should be added to the current states. After each of the new states is added, empty state transitions are completed for each of the current states until the size of the current states vector does not increase.

Test Programs and Inputs:

To verify the correctness of the simulation, I tested the program with the following NFA descriptions and tape inputs:

nfa1.txt with: nfa1-input.txt
nfa2.txt with: nfa2-input.txt
test-case1.txt with: 00101, 000000, 01011101, 10
test-case2.txt with: 000000000, 0, 010111, 1

Difficulties and Implementation Decisions:

The main difficulty in developing this simulator was parsing the NFA description files and inputs in C++. Though C++ does not have extended parsing capabilities, I chose to develop in C++ because of the experience I have had with the language.

A pivotal implementation decision was in the representation of the transition function. The data structure I used to store the transition function and use it to advance the NFA reflected the tabular representation presented in class with the formal definition of an NFA. It allowed for a straight-forward implementation of advancing the machine.