Humza Faraz and Joshua Philip Santos

November 1, 2021

CMSC 451

Sebald

Course Project Part 1 Report

Our task for the first part of the course project was implementing a Deterministic Finite Automata (DFA) on a string of input. Essentially, determining if user input is accepted into a state machine containing rules defined in the provided text file.

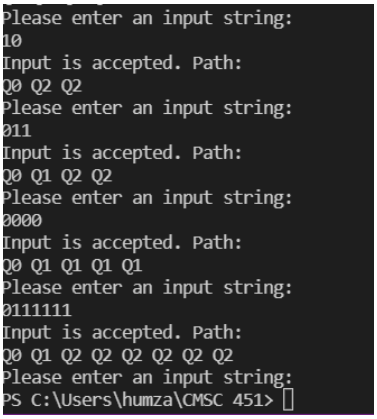
The way the text file is implemented is it provides the input alphabet, number of states, list of accept states, then the list of rules. The way the rules are formatted are for each line, consisting of the starting state, what character from the input alphabet to transition on, and the state to transition to. I wanted to keep the formatting of the way these variables are linked, so I stored my data for the rules in a multimap.

A multimap differs from a map because it can store multiple values for the same key. In this case each key would be a state in the machine and the values would be the different paths one could take from that state. To do this, my key would be a string type (for the start state) and the value would be a pair consisting of a char and string (for the input alphabet and the transition).

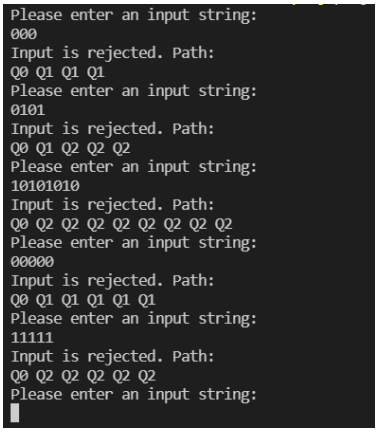
After validating the text file then loading in all my variables and rules, I prompt the user to enter an input string. I would then iterate through the string character by character, and find the next state based on the data stored in my multimap. I would then find the next state and add it

to my vector that keeps track of the path. After iteration is complete, I would then check if the state the user ends on is one of the accept states. I tell the user whether their input is accepted or rejected, then print the path taken. This process would continue until the user exits the program.

To test my code and make sure it was implemented properly, I did extensive testing on user validation and validation on the text file, then did several runs of my code. I tried having all states included in the accept states and testing if input would always be accepted:



Next, I tested if there were no accept states in the machine and checked if the input would always be rejected:



Following this and all the necessary validations, I believe the project has been implemented successfully. A guide for how to run and execute the simulator is included in the README file.

Sources:

https://www.csee.umbc.edu/~lsebald1/cmsc451-fa2021/project.shtml

https://www.geeksforgeeks.org/multimap-associative-containers-the-c-standard-template-library-stl/

https://www.cplusplus.com/doc/tutorial/files/

https://stackoverflow.com/questions/5990919/insert-pair-as-map-value

https://www.cplusplus.com/reference/map/multimap/begin/

https://www.tutorialspoint.com/multimap-find-in-cplusplus-stl

https://thispointer.com/finding-all-values-for-a-key-in-multimap-using-equals_range-example/