**CSC615M - Machine Project 3 Documentation**

Aquino, Kurt Neil

De La Salle University, Manila

254 Sitio Bagong Anyo, Pagsawitan, Santa Cruz, Laguna

(+63) 9273597974

kurt\_aquino@dlsu.edu.ph

1. **INTRODUCTION**

For the third learning output for the course of CSC615M – Automata Theory, Computability, Formal Languages, the students were tasked to design a system which would simulate the functionality of a Two Way Accepter (TWA) and determine whether a given input string will be accepted or rejected by the specified program.

As discussed in class, Two Way Acceptors are similar to the rest of the machines previously discussed, but are distinct in such a way that they can read the input string in either direction. We think of them as having a read head, which can move either left or right over the input string.

And although these machines seem to be much more powerful than one-way finite machines, having neither storage tape nor output tape proves that Two Way Acceptors are therefore no more capable than finite-state acceptors as language recognizers. Two Way Acceptors and finite-state acceptors are technically equivalent with one another in the sense that they only accept regular sets. This further shows that storage tapes are essential if the limitations of finite-state machines are to be overcome by tape automata (Denning, P.J, Dennis, J.B., & Qualitz, J.E., 1978).

Overall this project aims to simulate the process of accepting or rejecting an input strings in a given Two Way Acceptor (TWA).

1. **SOFTWARE DESIGN**
   1. **Inputs**

As the project aims to simulate the functionality of a Two Way Acceptor (TWA), the inputs to be accepted will be the machine’s corresponding components, which is formally defined by:

* Q = set of internal states
* T = set of tape symbols
* P = program containing set of instructions of the form
* q] scanleft(t,q’)
* q] scanright(t,q’)
* qi = initial state
* F = set of final states

Other than the machine’s components, another input to be accepted by the system is the input strings to be tested. For the user’s ease of access, the inputs will be saved in their own respective text files.

For the machine’s input program, the content of the text file will only contain the transition functions. From there, the system will derive the machine’s corresponding input alphabet as well as the unique states, including the start and accepting states.

Sample input program text file components:

start,right,b,start

start,right,a,2

start,right,#,3

2,right,a,start

2,right,b,2

2,right,#,4

4,left,a,4

4,left,b,4

4,left,#,5

5,right,a,5

5,right,b,6

5,right,#,accept

6,right,a,6

6,right,b,5

6,right,#,3

This program accepts an input string which contains an odd number of ‘a’ characters and an even number of ‘b’ characters, and is formally defined as:

For the machine’s input string, the content of the text file will contain a series of strings to be tested by the given program in order to determine if each string is either rejected or accepted.

Sample input strings text file components:

aaaaabb

aababb

aabbbbbbbbbbbbb

12121212

* 1. **Algorithm and Implementation**

In order to simulate the functionality of a Two Way Accepter, the corresponding implementation of each of the machine’s components should first be discussed.

* + 1. **File Reader**

This component of the system is mainly involved with the reading of the given text files and then storing their corresponding values to the machine’s components.

The file readers are separated into 2 parts: One for the input program, and another for the input strings.

As discussed in Section 2.1., the program text file will only contain the P (set of instructions) component of the Two Way Acceptor (TWA) as the rest of the components can be formally derived from there.

Given a(n) instruction(s) in the form of:

<state>, <scan\_direction>, <input\_symbol>, <next\_state>

* Q can be derived from the unique set of <states>
* T can be derived from the unique set of <input\_symbols>
* can be derived if a <state> is defined by the term “start”
* F can be derived from the unique set of <states> whose <next\_states> are defined as “accept”
* Instructions/transitions for P can be derived from the set of the corresponding <scan\_directions> of <states> defined by either “left” or “right”
  + 1. **States and Transition Objects**

This component focuses on the storing and manipulation of the states and their corresponding transitions into the system.

State Objects are essentially the program instructions of the machine and are defined in the system to have the following components:

* label – state name
* direction – scanleft or scanright
* transitions – set of transitions when given certain input symbols and their corresponding succeeding states
* isFinal – determines whether the state is an accepting state or not

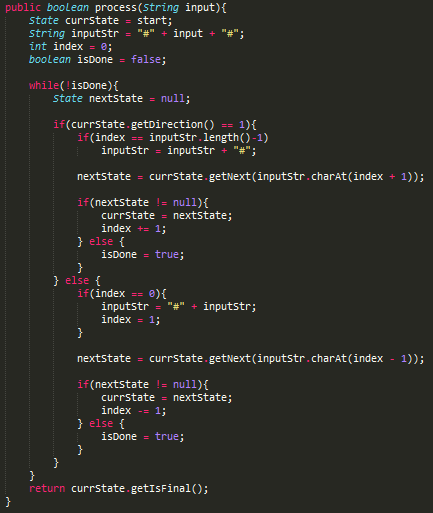
State objects within the system also provides the functionality of indicating the succeeding state to go to when given a specific input symbol.

Transition Objects are the set of transitions a state has when given a specific input symbol and are defined in the system to have the following components:

* symbol – input symbol
* nextState - State Object of the succeeding state given an input symbol
  + 1. **Two Way Acceptor**

This component of the system simulates to process of performing the functionality of a Two Way Acceptor given its corresponding state of instructions and input string.

The function essentially scans through each symbol of the given string, and proceeds to their corresponding states. The direction on which the header proceeds through the input string depends on the succeeding state of the current state given a scanned symbol as well as the scan direction assigned to the current state. The program will then form a conclusion once the entire input string has been read, according to the sequence of instructions performed, and determine whether the last state where the acceptor landed is an accepting state or not.



*Figure 2.1. Code for performing TWA*

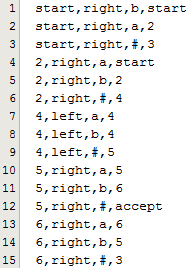
*testing given an input string*

* 1. **Output**

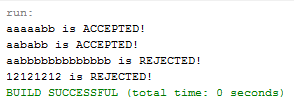
The system outputs a statement whether the given input strings are either rejected or accepted. It is also optional to view the sequence of instructions performed before forming a conclusion. The sequence includes presenting the current state in the sequence as well as the set of transitions available for that given state.

1. **TEST CASES**
   1. **Odd number of a’s, Even number of b’s**

Formal Definition:

**

*Figure 3.1. Test Case 1 Program*

**

*Figure 3.2. Test Case 1 Outputs*

* 1. **2.2 consecutive a’s with 3 b’s and 4 c’s in before each succeeding a**

1. **CONCLUSION**

Overall, the system was successfully implemented. As proven by the test cases of varying scenarios, the system can successfully simulate the functionality of a Two Way Acceptor (TWA).

By testing out various input strings which covers some of the various scenarios on the Two Way Acceptors discussed in class, it can be concluded that the system works as according to its implementation.

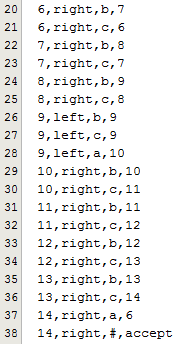
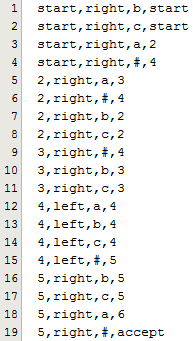
This system will also be useful for the succeeding machine project as it involves setting up Turing Machines.

1. **REFERENCES**

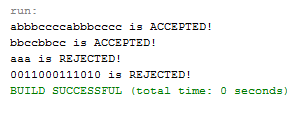
Denning, P.J., Dennis, J.B., & Qualitz, J.E. (1978). *Machines, languages, and computation*. New Jersey: Prentice Hall.

1. **ASSESSMENT RUBRIC**

|  |  |
| --- | --- |
| **Criteria** | **Score** |
| Input-Output Modules | **30** |
| Core Process | **30** |
| Quality of testing | **19** |
| Documentation | **18** |



*Figure 3.3. Test Case 2 Program*



*Figure 3.4. Test Case 2 Outputs*