CS 410 Automata Theory and Formal Languages

Project 3 Final Report

Bora Özügüzel

S014465

1. Introduction

In this homework, a project that reads a text file of a Deterministic Turing Machine, and then prints the route where each state that the Turing Machine visited and the result that if the string is accepted or rejected is printed out. If the Turing Machine enters in a infinite loop, it should print looped as a result.

The project is in Java.
The input file format for the project is given as a txt file. The first line is the number of variables in the input alphabet, which is given as a single line where the variables are separated by space. The next two lines are for tape alphabet and their format is the same as the input alphabet. The next line is the blank symbol. The next two lines are for the states and this format is same as the input alphabet, but unlike input alphabet to tape alphabet a state can have more than one character. The next three lines are the start state accept state and the reject state. The rest of the lines until the last line are transitions. The first part is the state transition starts from and the last part is the state transition ends. The third part is the new symbol that replaces the second part, which is the old symbol, in the tape of the Turing machine that the head is pointing to. Then the head moved according the fourth part where L is left and R is right. The last line is the string input of The Turing Machine.

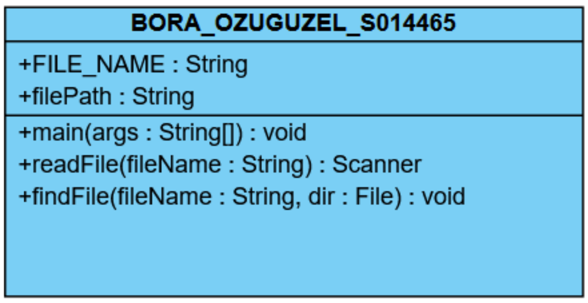
The example input format given in the project description:

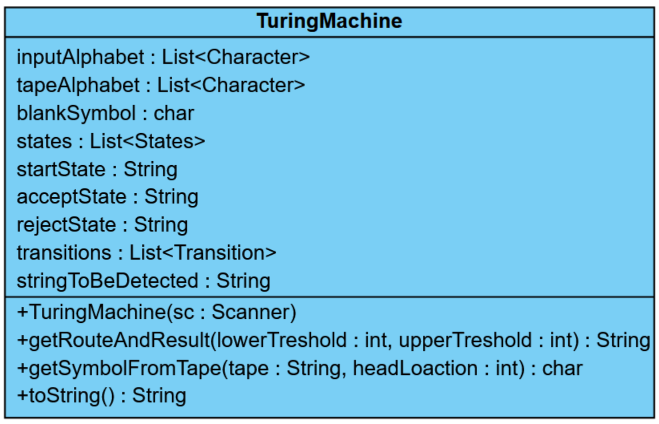
```
1
0
2
0 X
b
7
q1 q2 q3 q4 q5 qA qR
q1
qA
qR
```

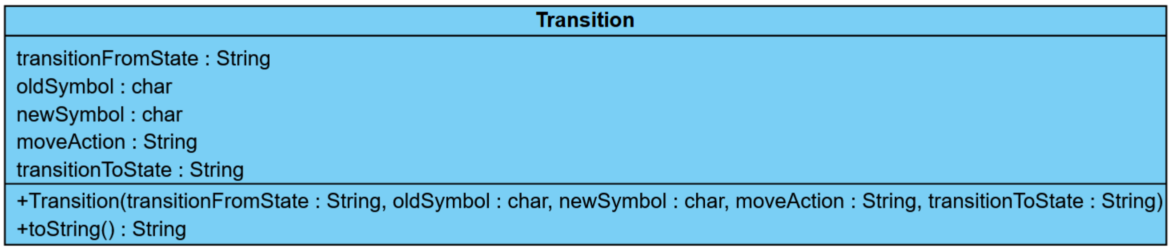
```
q1 0 b R q2
q1 b b R qR
q1 X X R qR
q2 0 X R q3
q2 X X R q2
q2 b b R qA
q3 X X R q3
q3 0 0 R q4
q3 b b L q5
q4 X X R q4
q4 0 X R q3
q4 b b R qR
q5 0 0 L q5
q5 X X L q5
q5 b b R q2
000
```

2. Classes and Methods

Class Diagram of the project is given below:


BORA_OZUGUZEL_S014465
+FILE_NAME : String
+filePath : String
+main(args : String[]) : void
+readFile(fileName : String) : Scanner
+findFile(fileName : String, dir : File) : void


TuringMachine
inputAlphabet : List<Character>
tapeAlphabet : List<Character>
blankSymbol : char
states : List<States>
startState : String
acceptState : String
rejectState : String
transitions : List<Transition>
stringToBeDetected : String
+TuringMachine(sc : Scanner)
+getRouteAndResult(lowerTreshold : int, upperTreshold : int) : String
+getSymbolFromTape(tape : String, headLoaction : int) : char
+toString() : String


Transition
transitionFromState : String
oldSymbol : char
newSymbol : char
moveAction : String
transitionToState : String
+Transition(transitionFromState : String, oldSymbol : char, newSymbol : char, moveAction : String, transitionToState : String)
+toString() : String

3. Implementation Details

In the main method, a method to read the Input_BORA_OZUGUZEL_S014465.txt file will be called named readFile.

The method tries to find the file in the project folder with the given filename defined in FILE_NAME which is assumed to be Input_BORA_OZUGUZEL_S014465.txt. If it cannot find it directly, it will try to

search the file in the whole project with the method named findFile. This file will be accessed by the Scanner class from the Java.Util Package. The result will be sent to instantiate a new TuringMachine object. The constructor is shown as bellow:

public TuringMachine(Scanner sc) {…}

The scanner is parsed by each line. The first line is put inside int value numberOfInputVaribles. The second line, which holds the input alphabet information is tokenized. Each token is added to a Character ArrayList named inputAlphabet in a while loop which is broken by the numberOfInputVaribles. The next line and the line after that holds the information of tape variables which is assigned to a Character ArrayList named tapeAlphabet similar to the input alphabet. The next line holds a char value called blankSymbol. The next line and the line after that holds the information of the states, which is assigned to a String ArrayList in a similar way to the tape alphabet. The next three lines are string values startState, acceptState and rejectState. Next lines until the last line are the transitions. In a while loop, in an if clause it is checked that if there are more lines after the current one. If there are no more lines, the current line is assigned to a string value called stringToBeDetected. If there are more lines, each line is tokenized to be assigned to a new object called Transition. The constructor is shown as below:

public Transition(String transitionFromState, char oldSymbol, char newSymbol, String moveAction, String transitionToState) {…}

The transition has five values. The first one is a string value transitionFromState, which shows which state the transition is coming from. The third, newSymbol, is used to replace the second, oldSymbol, in the tape of the Turing Machine. The fourth String value which can be either L or R, is used to move the head of the Turing Machine left or right on the tape. The last value, transitionToState, is used to change the states.

Two int values called lowerTreshold and upperTreshold with values 35 and 100 are sent to a method called getRouteAndResult, which prepares a String, so it can be printed in the main method. The method is shown as below:

public String getRouteAndResult(int lowerTreshold, int upperTreshold) {…}

In the getRouteAndResult, an String ArrayList called route and a String called tmResult are created to later assign the desired values. The value of stringToBeDetected is copied to a String value called tape. The value of startState is copied to a string called currentState.
The initial length of tape is also stored in a string initialTapeLength. A String ArrayList called history is created to hold the history of Turing machine.

A while true loop is created. The history of the current iteration is created as a concatenation of the tape, the currentState and the headLocation.

If the currentState is the same as acceptState or the rejectState the loop breaks, after assigning accepted or rejected to tmResult and adding the currentState to route. If currentHistory is already exists in history and history is larger than lowerTreshold or history is the same as upperTreshold it is deemed likely that the machine loops by having the same instance of tape, currentStae and headLocation. If the length of tape increased from the initialTapeLength by the multiplication of size of tapeAlphabet and states while also being larger than the lowerTreshold, it is deemed that the transitions of the states are looped and the headLocation and the tape are getting larger and larger. Then the currentState is added to the route and tmResult is assigned looped before the break of the while loop.

currentHistory is added to history.

A method called getSymbolFromTape is needed, which reads the current symbol at the tape pointed by headLocation. If the tape is empty or the head points to symbol that is located more to the right than the entire string blankSymbol is returned. Otherwise, the value at the index headLocation is returned from tape. The method is shown as below:

```
public char getSymbolFromTape(String tape, int headLocation) {…}
```

In a Java Stream lambda expression the first transition where the transitionFromState is equal to the currentState and the oldSymbol is equal to the value returned from getSymbolFromTape is as optionalTransition. If the optionalTransition is empty the currentState is added to route and the tmResult is rejected. But this can only happen when the transitions are missing from the Turing Machine, which cannot happen in a Deterministic Turing Machine. If it is present it is assigned to currentTransition;

If the headLocation points to the blank symbol next to the tape, the new symbol from the currentTransition is appended to the tape. Otherwise, the tape is split with substring where the index that is pointed by headLocation is missing and the newSymbol is concatenated between.

If moveAction is L, headLocation is reduced by one, but if the headLocation is zero the it stays at the same place. If moveAction is R, headLocation is incremented.

Before the end of the while loop, the transitionToState is copied to the currentState.

After the while loop the route and the result is assigned to the return value of getRouteAndResult.

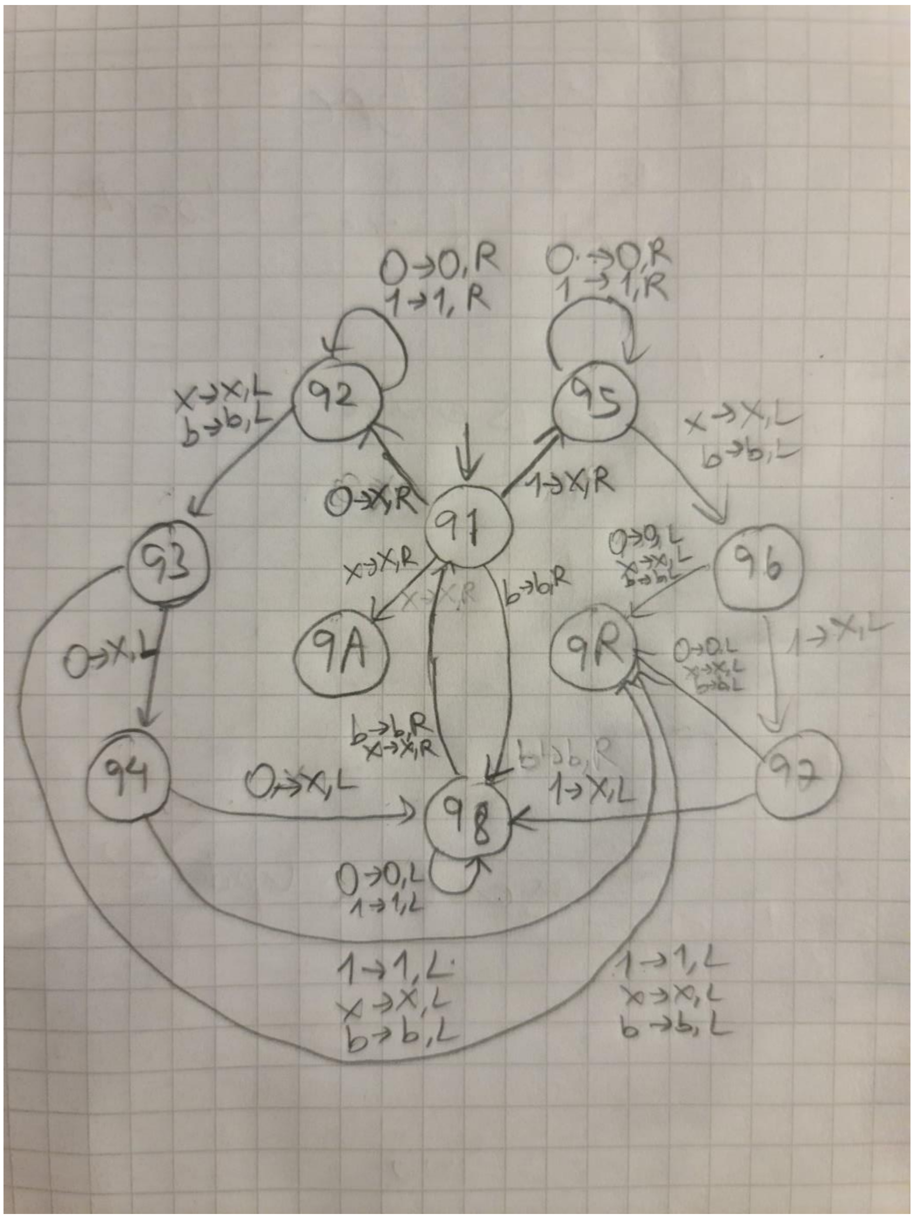
4. Results

Input_BORA_OZUGUZEL_S014465.txt

Description:

In TM1 there are two symbols in the input alphabet and the tape alphabet is the same, but it has an additional X. The strings accepted by the machine are like a palindrome, but the symbols on the right side occur twice. The length of the string is at least three. An empty string causes a loop. The symbols after the first blank symbol are ignored.

{wz | wz is part of the string until the first blank symbol. The length of z is two times of w. The i'th index of w is equal to the symbols in (2i)'th and (2i-1)'th index from the right side of z. The length of wz is at least three.}



Input_BORA_OZUGUZEL_S014465 input from file:

```
2
0 1
3
0 1 X
```

```
b
10
q1 q2 q3 q4 q5 q6 q7 q8 qA qR
q1
qA
qR
q1 0 X R q2
q1 1 X R q5
q1 X X R qA
q1 b b R q8
q2 0 0 R q2
q2 1 1 R q2
q2 X X L q3
q2 b b L q3
q3 0 X L q4
q3 1 1 L qR
q3 X X L qR
q3 b b L qR
q4 0 X L q8
q4 1 1 L qR
q4 X X L qR
q4 b b L qR
q5 0 0 R q5
q5 1 1 R q5
q5 X X L q6
q5 b b L q6
q6 0 0 L qR
q6 1 X L q7
q6 X X L qR
q6 b b L qR
q7 0 0 L qR
q7 1 X L q8
q7 X X L qR
q7 b b L qR
q8 0 0 L q8
q8 1 1 L q8
q8 X X R q1
q8 b b R q1
101110011
```

Input_BORA_OZUGUZEL_S014465 output to console with string 101110011:

```
ROUT: q1 q5 q5 q5 q5 q5 q5 q5 q5 q5 q6 q7 q8 q8 q8 q8 q8 q8 q8 q1 q2 q2 q2 q2 q2
q2 q3 q4 q8 q8 q8 q8 q1 q5 q5 q5 q6 q7 q8 q1 qA
```

```
RESULT: accepted
```

Input_BORA_OZUGUZEL_S014465 output to console with string 1010011:

```
ROUT: q1 q5 q5 q5 q5 q5 q5 q5 q6 q7 q8 q8 q8 q8 q8 q1 q2 q2 q2 q2 q3 q4 q8 q8 q1
q5 q6 qR
RESULT: rejected
```

Input_BORA_OZUGUZEL_S014465 output to console with string b:

```
ROUT: q1 q8 q1 q8 q1 q8 q1 q8 q1 q8 q1 q8 q1 q8 q1 q8 q1 q8 q1 q8 q1 q8 q1 q8 q1
q8 q1 q8 q1 q8 q1 q8 q1 q8 q1 q8 q1
RESULT: looped
```