REVISION HISTORY

Date	Version	Description	Author
15/10/2021	0.1	Design of the Sections	Muhammed Esad Simitcioglu
16/10/2021	0.5	Diagrams added	Muhammed Esad Simitcioglu
17/10/2021	1.0	Details added	Muhammed Esad Simitcioglu
05/11/2021	1.1	Interface and Descriptions are added	Muhammed Esad Simitcioglu

TABLE OF CONTENTS

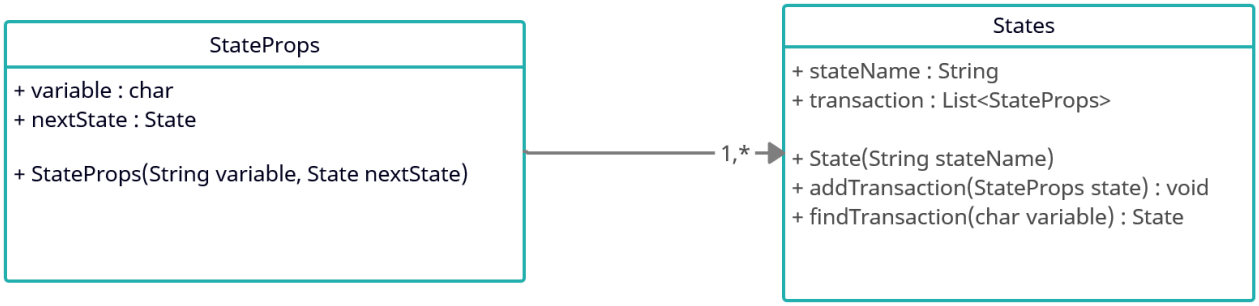
Table of Contents

1 Introduction

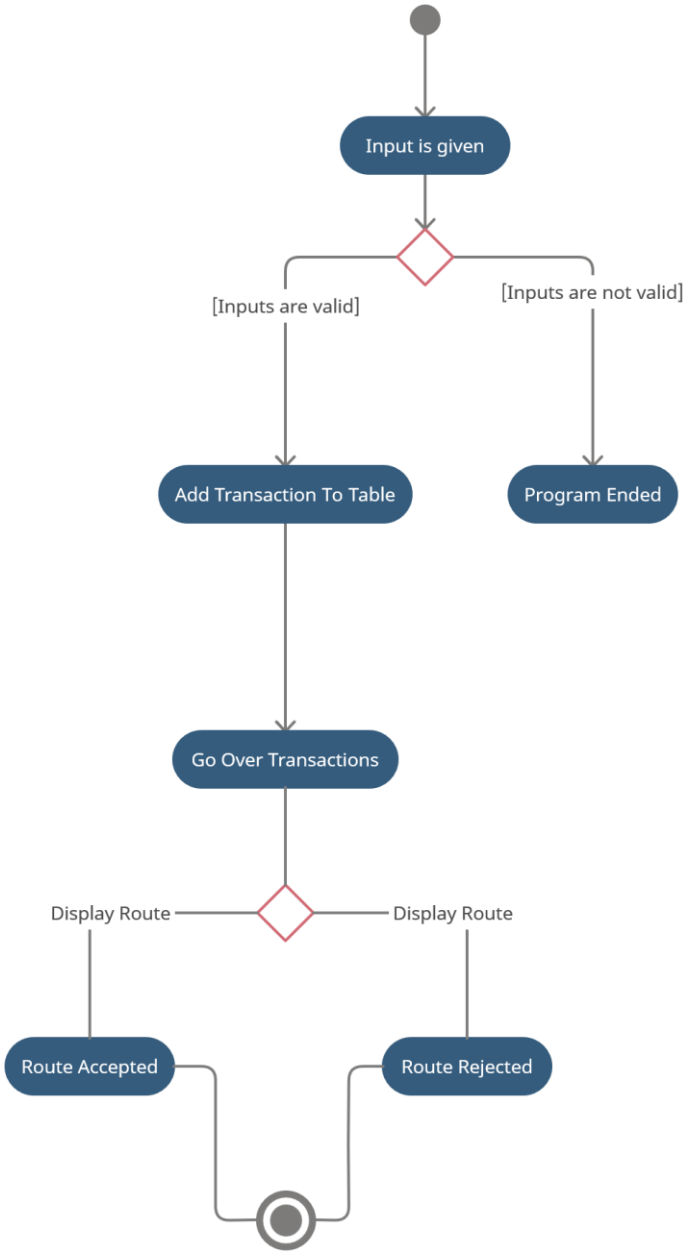
This document describes the design of the Deterministic Finite Automate (DFA). This program should be working exactly as a DFA would do. Given a string the simulated DFA should be able to tell if the string is accepted or rejected. This program should output the information about the given string being accepted or rejected as well as which states it visited.

2 Software Architecture overview

Class diagram is given below.



The Activity Diagram is given below:



3 Software design description

This program will not be desktop application and no need for database. So, in that case there won't be a UI or database implementation. The input and output format of the program should be like in the below.

Input:
2 (number of states)
2 (number of variables)
1 (number of goal states)
q1 q2 (states)
q2 (goal state(s))
a b (variables)
q1 a q1 (from q1 with a to q1)
q1 b q2 (from q1 with b to q2)
q2 a q2 (from q2 with a to q2)
q2 b q1 (from q2 with b to q1)
aba (string to be detected)
abababaa (string to be detected)

Output:
q1 q2 q2 (route taken)
Accepted
q1 q2 q2 q1 q1 q2 q2 q1 q1 (route taken)
Rejected

The program will construct a List of “State” object to store all of the given states. After constructing and connecting “States”, the program will create transaction table for each “State”. Each "State" will know where to go when the appropriate variable arrives. The name of each "State" will be printed to reveal the route the program took. Finally, the program will print "Accepted" if the last "State" is one of the target states. Otherwise the program will print “Rejected”.

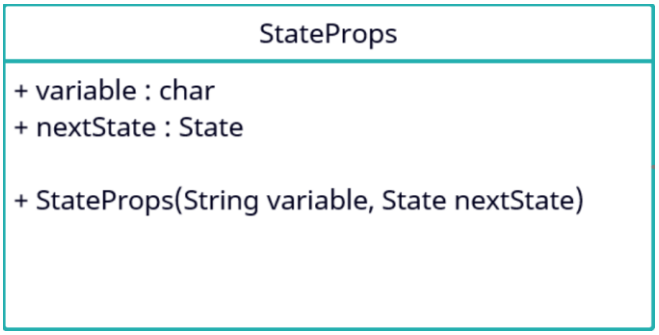
3.1 StateProps

3.1.1 Component Interface

Methods of State which are available from other components are:

public StateProps(String variable, State nextState) {} – Creates the State properties as specific variable and its next state.

3.1.2 Component Design Description



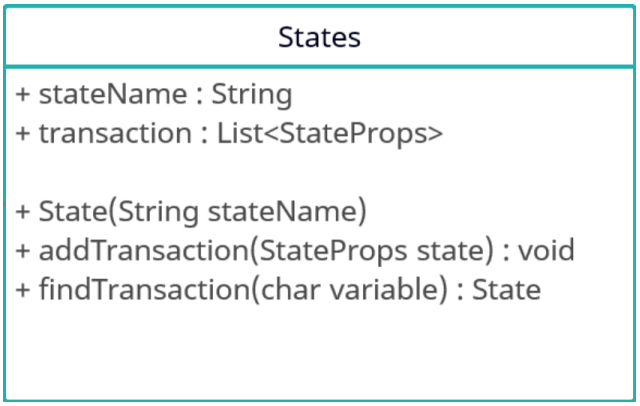
3.2 State

3.2.1 Component Interface

Methods of State which are available from other components are:

public void addTransaction (StateProps state) {} – Add variable transactions to the specific state.
public State findTransaction (char variable) {} – Return the next state for the specific variable.

3.2.2 Component Design Description



4 COTS Identification

COTS (Commercial off the shell) libraries used :

Not Applicable