CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,

BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A Mini Project Report
on

"DFA ALGORITHM"

Submitted in Partial fulfillment of the Requirements for the VI Semester of the Degree of

Bachelor of Engineering
In
Computer Science & Engineering

By
DIVYA T
(1CR15CS058)

Under the Guidance of

Mrs. Sagarika Behera
Assoc Professor, Dept. of CSE

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

List of Figures

List of Tables

ABSTRACT

A finite automato (FA) is a simple idealized machine used to recognize patterns within input taken from some character set (or alphabet) C. The job of an FA is to *accept* or *reject* an input depending on whether the pattern defined by the FA occurs in the input.

A finite automaton consists of:

- a finite set S of N states
- a special start state
- a set of final (or accepting) states
- a set of transitions T from one state to another, labeled with chars in C

Finite automaton can be classified into two types

1. Deterministic Finite Automata(DFA)
2. Non-Deterministic Finite Automata(NFA)

A DFA operates in the following manner: when program starts the current state is assumed to be the initial state q0, on each input symbol or character the current state is supposed to move on another state (including itself). When the string reach the most right symbol (last one) the string is accepted only if the current state is one of the accepted states, otherwise the string is rejected. The abbreviation DFA for deterministic finite automata is called deterministic because on each state and input character a unique transition is defined. The deterministic finite automaton is the only way so far to execute an automaton, that means that only a DFA can be implemented.

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without mentioning the people whose proper guidance and encouragement has served as a beacon and crowned my efforts with success. We take an opportunity to thank all the distinguished personalities for their enormous and precious support and encouragement throughout the duration of this project work.

We take this opportunity to express our sincere gratitude and respect to **CMR Institute of Technology, Bangalore** for providing us an opportunity to carry out our project work.

We have a great pleasure in expressing our deep sense of gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore for his constant encouragement.

We sincerely thank **Prof. Jhansi Rani**, HOD, Department of Computer Science and Engineering, CMR Institute of Technology for the immense support given to us. Her incessant encouragement and invaluable technical support have been of immense help in realizing this project work.

With profound sense of gratitude, we acknowledge the guidance and support extended by **Prof. Sagarika Behera**, Assistant professor, Department of Computer science and Engineering, CMRIT, Bangalore. Her guidance gave us the environment to enhance our knowledge, skills and to reach the pinnacle with sheer determination, dedication and hard work..

We also extend our thanks to the faculty of Computer Science and Department who directly or indirectly encouraged us throughout the course of project work.

We thank our parents and friends for all their moral support they have given us during the completion of this work.

Above all, we thank the Lord Almighty for his grace on us to succeed in this endeavour.

- Divya T

CHAPTER 1

INTRODUCTION

1.1 Aim

Write a C++/java program to implement a DFA which accepts the strings that contains at most 3 a's. The program uses a class for DFA. It gives the output as "yes" if it accepts, otherwise "NO".

1.2 Description

In the theory of computation, a branch of theoretical computer science, a deterministic finite automaton (DFA)—also known as a deterministic finite acceptor (DFA) and a deterministic finite state machine (DFSM) or a deterministic finite state automaton(DFSA)—is a finite-state machine that accepts or rejects strings of symbols and only produces a unique computation (or run) of the automaton for each input string. *Deterministic* refers to the uniqueness of the computation. In search of the simplest models to capture finite-state machines, Warren McCulloch and Walter Pitts were among the first researchers to introduce a concept similar to finite automata in 1943.

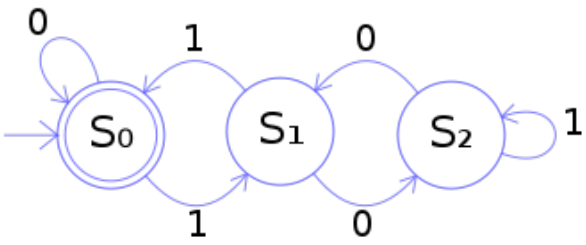

Figure 1.1.1 DFA

The figure illustrates a deterministic finite automaton using a state diagram. In the automaton, there are three states: S_0, S_1, and S_2 (denoted graphically by circles). The automaton takes a finite sequence of 0s and 1s as input. For each state, there is a transition arrow leading out to a next state for both 0 and 1. Upon reading a symbol, a DFA jumps *deterministically* from one state to another by following the transition arrow. For example, if the automaton is currently in state S_0 and the current input symbol is 1, then it deterministically jumps to state S_1. A DFA has a *start state* (denoted graphically by an arrow coming in from nowhere) where computations begin, and a set of *accept states* (denoted graphically by a double circle) which help define when a computation is successful.A DFA is defined as an abstract mathematical concept, but is often implemented in hardware and software for solving various specific problems. For example, a DFA can model software that decides whether or not online user input such as email addresses are valid. DFAs recognize exactly the set of regular languages, which are, among other things, useful for doing lexical

analysis and pattern matching. DFAs can be built from nondeterministic finite automata (NFAs) using the power set construction method.

CHAPTER 2

SYSTEM REQUIREMENTS

2.1 Hardware Requirements

CPU Speed	1.5 GHz or higher
Processor	Intel Pentium® or higher
Memory/RAM	2 GB or higher (32-bit) 8 GB or higher (64-bit)
Display Properties	24-bit colour depth
Screen Resolution	1024 x 768 or higher at normal size (96 dpi)
Swap Space	Determined by the operating system, 500 MB or higher
Disk Space	1 GB for installing the Platform and Synthesis Applications
Video/Graphics Adapter	64 MB RAM or higher

Table 2.1.1 : Hardware Requirements

2.2 Software Requirements

Platform	Windows, Ubuntu
SDK	J2SE SDK
Java Version	1.4.0 or greater
IDE	Eclipse IDE
Java Extension	JFLAP

Table 2.2.1 : Software Requirements

CHAPTER 3

DESIGN

3.1 Algorithm for DFA

A deterministic finite automaton M is a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, consisting of

- a finite set of states (Q)
- a finite set of input symbols called the alphabet (Σ)
- a transition function $(\delta : Q \times \Sigma \rightarrow Q)$
- an initial or start state ($q_0 \in Q$)
- a set of accept states ($F \subseteq Q$)

Let $w = a_1a_2 \ldots a_n$ be a string over the alphabet Σ. The automaton M accepts the string w if a sequence of states, $r_0, r_1, \ldots, r_n$, exists in Q with the following conditions:

1. $r_0 = q_0$
2. $r_{i+1} = \delta(r_i, a_{i+1})$, for $i = 0, \ldots, n-1$
3. $r_n \in F$.

In words, the first condition says that the machine starts in the start state q_0. The second condition says that given each character of string w, the machine will transition from state to state according to the transition function δ. The last condition says that the machine accepts w if the last input of w causes the machine to halt in one of the accepting states. Otherwise, it is said that the automaton rejects the string. The set of strings that M accepts is the language recognized by M and this language is denoted by L(M).

A deterministic finite automaton without accept states and without a starting state is known as a transition system or semi automation.

3.2 Graphical Representation of a DFA

A DFA is represented by digraphs called state diagram.

- The vertices represent the states.
- The arrows labelled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.
- The final state is indicated by double circles.

CHAPTER 4

IMPLEMENTATION

4.1 Solution

1) Consider the alphabet to be ∑={a,b}
2) The solution should be able to accept a string with any number of 'b's
3) The solution can accept a string that contains only one 'a', two 'a's or three 'a's along with any number of 'b's. The number of 'b's can be zero. The 'a'/'a's can occupy any position in the string.
4) The solution is as follows:

   a)Assume the states are Q={q0,q1,q2,q3,qError}

   b)The q0 will be the start state.q0,q1,q2 and q3 are the final states.

   c) After entering the start state, at q0, when it encounters 'b', it stays in state q0. When it encounters 'a', it goes to state q1.

   d) At state q1, when it encounters 'b', it stays in state q1. When it encounters 'a', it goes to state q2.

   e) At state q2, when it encounters 'b', it stays in state q2. When it encounters 'a', it goes to state q3.

   f) At state q3, when it encounters 'b', it stays in state q3. When it encounters 'a', it goes to the qError state because it will encounter the 4th 'a'. According to the problem statement, it can accept at most 3 'a's.

4.2 Source code

```
import java.util.Scanner;
public class DFAInput {
        public static void main(String[] args) {
                Scanner s= new Scanner(System.in);
                System.out.println("Enter the string:");
```

```
                String input=s.nextLine();
                DFA obj=new DFA();
                int count= obj.Input(input);;
                if(count>3)
                        System.out.println("Invalid input string.The number of a's exceed 3.");
                else
                        System.out.println("Valid input string.");
        }
}

public class DFA {
        public int Input(String input) {
                int counter = 0;
                for( int i=0; i<input.length(); i++ ) {
                   if( input.charAt(i) == 'a' )
                       counter++;
                 }
                return counter;
        }
}
```

4.3 Output Snapshots

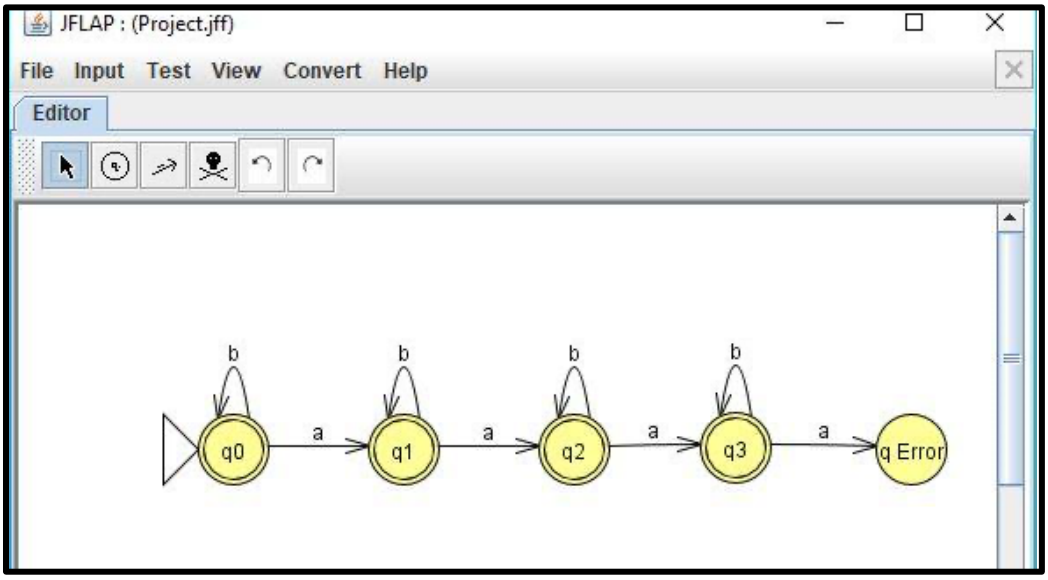

Figure 4.3.1: JFLAP implementation

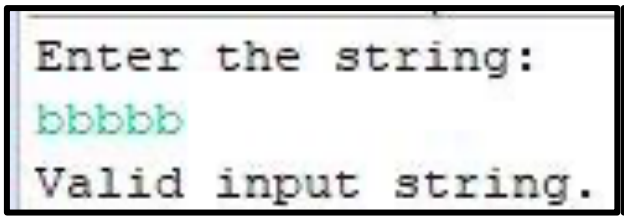

Figure 4.3.2 Java Output for string with no a's

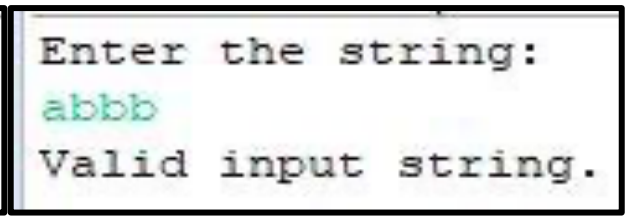

Figure 4.3.3 Java Output for string with 1 a

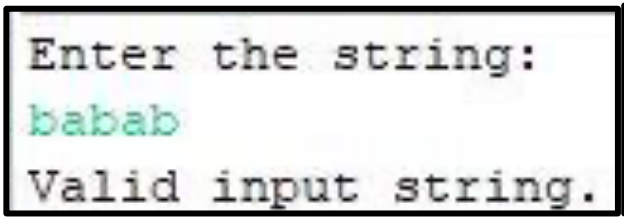

Figure 4.3.2 Java Output for string with 2 a's

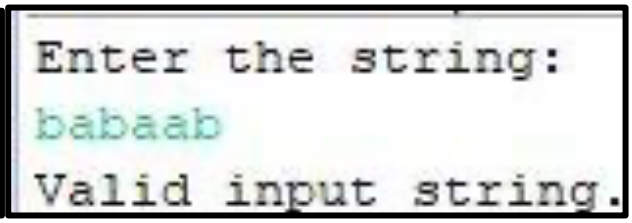

Figure 4.3.2 Java Output for string with 3 a's

CHAPTER 5

CONCLUSION

DFAs are one of the most practical models of computation, since there is a trivial linear time, constant-space, online algorithm to simulate a DFA on a stream of input. Also, there are efficient algorithms to find a DFA recognizing:

- the complement of the language recognized by a given DFA.
- the union/intersection of the languages recognized by two given DFAs.

Because DFAs can be reduced to a *canonical form* (minimal DFAs), there are also efficient algorithms to determine:

- whether a DFA accepts any strings
- whether a DFA accepts all strings
- whether two DFAs recognize the same language
- the DFA with a minimum number of states for a particular regular language

DFAs are equivalent in computing power to nondeterministic finite automata (NFAs). This is because, firstly any DFA is also an NFA, so an NFA can do what a DFA can do. Also, given an NFA, using the powerset

construction one can build a DFA that recognizes the same language as the NFA, although the DFA could have exponentially larger number of states than the NFA.

On the other hand, finite state automata are of strictly limited power in the languages they can recognize; many simple languages, including any problem that requires more than constant space to solve, cannot be recognized by a DFA. The classic example of a simply described language that no DFA can recognize is bracket or Dyck language, i.e., the language that consists of properly paired brackets such as word "(()())". Intuitively, no DFA can recognize the Dyck language because DFAs are not capable of counting: a DFA-like automaton needs to have a state to represent any possible number of "currently open" parentheses, meaning it would need an unbounded number of states. Another simpler example is the language consisting of strings of the form $a^n b^n$ for some finite but arbitrary number of a's, followed by an equal number of b's.

BIBLIOGRAPHY

[1] Google for problem solving

[2] Elaine Rich, Automata, Computability and Complexity, 1st Edition, Pearson Education,2012/2013

[3] K L P Mishra, N Chandrasekaran , 3rd Edition, Theory of Computer Science, PhI, 2012.

[4] Herbert Schildt, Java The Complete Reference, 7th Edition, Tata McGraw Hill, 2007.