OverView

Deterministic Finite Automata

In DFA, for each input symbol, one can determine the state to which the machine will move. Hence, it is called **Deterministic Automaton**. As it has a finite number of states, the machine is called **Deterministic Finite Machine** or **Deterministic Finite Automaton.**

Formal Definition

A DFA can be represented by a 5-tuple (Q, ∑, δ, q0, F) where −

* **Q** is a finite set of states.
* **∑** is a finite set of symbols called the alphabet.
* **δ** is the transition function where δ: Q × ∑ → Q
* **q0** is the initial state from where any input is processed (q0 ∈ Q).
* **F** is a set of final state/states of Q (F ⊆ Q).

Graphical Representation of a DFA

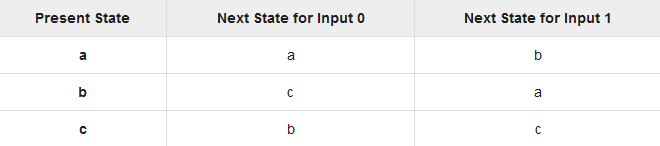
A DFA is represented by digraphs called **state diagram**.

* The vertices represent the states.
* The arcs labeled 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.

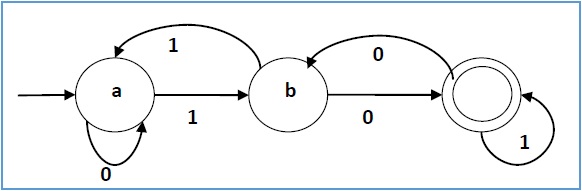
**Example**

Let a deterministic finite automaton P be P = ( Q, ∑, q0, F), where Q = {a, b, c}, ∑ = {0, 1}, q0 = {a}, F = {c}.

Transition function δ as shown by the following table



Its graphical representation would be as follows

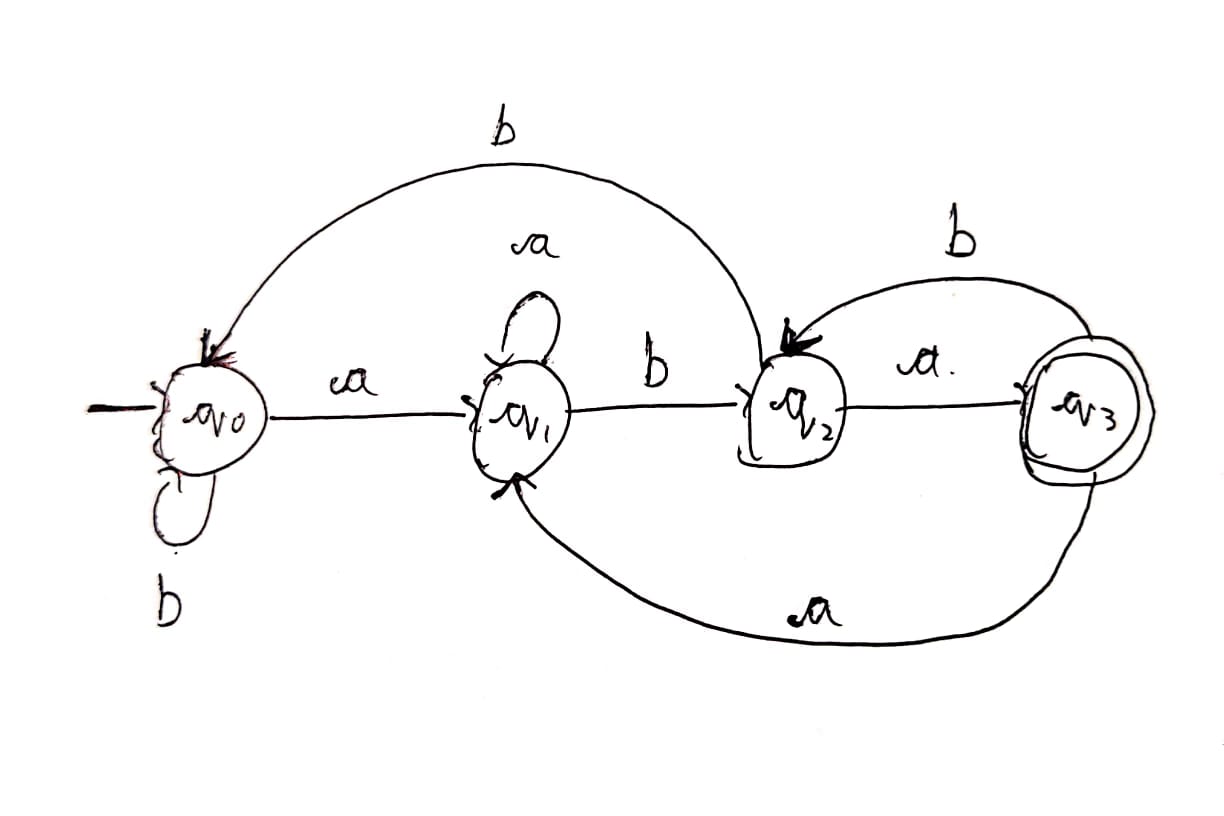


Deterministic Finite Automaton To Accept Strings Ending With **aba**

**State diagram** for a Deterministic Finite Automaton designed to accept strings ending with **aba** is as follows:

|  |  |  |
| --- | --- | --- |
| **Present State** | **Input** | **Next State** |
| q0 | b | q0 |
| q0 | a | q1 |
| q1 | b | q2 |
| q1 | a | q1 |
| q2 | b | q0 |
| q2 | a | q3 |
| q3 | a | q1 |
| q3 | b | q2 |

**Transition Diagram**



Source Code

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

int dfa = 0;

void **q0**(char c){

if(c == 'b')

dfa = 1;

else if(c == 'a')

dfa = 0;

else

dfa = -1;

}

void **q1**(char c){

if(c == 'a')

dfa = 1;

else if(c == 'b')

dfa = 2;

else

dfa = -1;

}

void **q2**(char c){

if(c == 'a')

dfa = 3;

else if(c == 'b')

dfa = 0;

else

dfa = -1;

}

void **q3**(char c){

if(c == 'a')

dfa = 1;

else if(c == 'b')

dfa = 2;

else

dfa = -1;

}

int **isAccepted**(char str[]){

int i, len = **strlen**(str);

for (i = 0; i < len; i++){

if (dfa == 0)

**q0**(str[i]);

else if (dfa == 1)

**q1**(str[i]);

else if (dfa == 2)

**q2**(str[i]);

else if (dfa == 3)

**q3**(str[i]);

else

return 0;

}

if(dfa == 3)

return 1;

else

return 0;

}

int **main**(){

char \*str;

int size, option;

**printf**("Automata To Accept Strings Ending With aba\n\n");

while(1){

**printf**("1. Check New String\n2. Exit\n");

**printf**("Enter Option\n");

**scanf**("%d", &option);

switch(option){

case 1:{

**printf**("\nEnter Size of String:\n");

**scanf**("%d", &size);

str = (char\*)**malloc**(sizeof(char));

**printf**("Enter String:\n");

**scanf**("%s", str);

if(**isAccepted**(str))

**printf**("ACCEPTED\n");

else

**printf**("NOT ACCEPTED\n");

**printf**("\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

**free**(str);

break;

}

case 2:{

**exit**(0);

}

default:{}

}

}

return 0;

}

Output

Harshits-MacBook-Air-2:Desktop harshitkaisare$ cc TOCaba.c

Harshits-MacBook-Air-2:Desktop harshitkaisare$ ./a.out

Automata To Accept Strings Ending With aba

1. Check New String

2. Exit

Enter Option

1

Enter Size of String:

3

Enter String:

aba

ACCEPTED

\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Check New String

2. Exit

Enter Option

1

Enter Size of String:

5

Enter String:

ababa

ACCEPTED

\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Check New String

2. Exit

Enter Option

1

Enter Size of String:

5

Enter String:

aabab

NOT ACCEPTED

\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Check New String

2. Exit

Enter Option

1

1. Check New String

2. Exit

Enter Option

2

Harshits-MacBook-Air-2:Desktop harshitkaisare$

References

<https://www.geeksforgeeks.org/program-to-construct-a-dfa-which-accept-the-language-l-anbm-n-mod-20-m1/>

Project Repository

<https://github.com/harshithkaisare/TOC-Project>