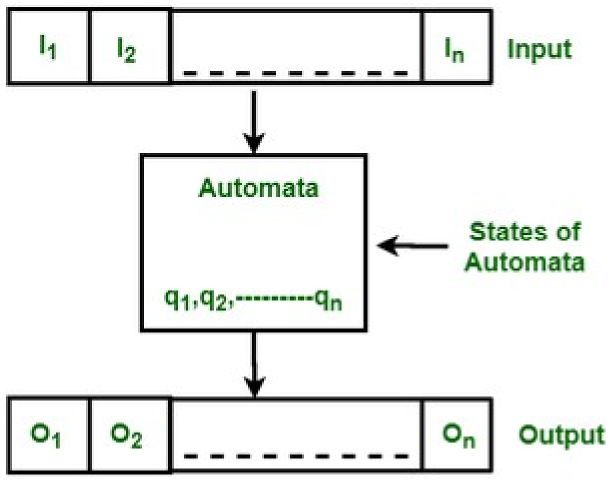
UNIT-1

FINITE AUTOMATA

Introduction to Finite Automata:

Finite Automata(FA) is the simplest machine to recognize patterns. The finite automata or finite state machine is an abstract machine that has five elements or tuples. It has a set of states and rules for moving from one state to another but it depends upon the applied input symbol. Basically, it is an abstract model of a digital computer. The following figure shows some essential features of general automation.



Finite Automata: It is used to recognize patterns of specific type input. It is the most restricted type of automata which can accept only regular languages (languages which can be expressed by regular expression using OR (+), Concatenation (.), Kleene Closure(*) like a*b*, (a+b) etc.)

Deterministic FA and Non-Deterministic FA: In deterministic FA, there is only one move from every state on every input symbol but in Non-Deterministic FA, there can be zero or more than one move from one state for an input symbol.

Note:

- Language accepted by NDFA and DFA are same.
- Power of NDFA and DFA is same.
- No. of states in NDFA is less than or equal to no. of states in equivalent DFA.
- For NFA with n-states, in worst case, the maximum states possible in DFA is 2n
- Every NFA can be converted to corresponding DFA.

The above figure shows the following features of automata:

Input

Output

States of automata

State relation

Output relation

A Finite Automata consists of the following:

Q : Finite set of states.

Σ : set of Input Symbols.

q : Initial state.

F : set of Final States.

δ : Transition Function.

1) Deterministic Finite Automata (DFA):

DFA consists of 5 tuples {Q, Σ, q, F, δ}.

Q : set of all states.

Σ : set of input symbols. (Symbols which machine takes as input)

q : Initial state. (Starting state of a machine)

F : set of final state.

δ : Transition Function, defined as δ : Q X Σ --> Q.

In a DFA, for a particular input character, the machine goes to one state only.

A transition function is defined on every state for every input symbol. Also in DFA null (or ε) move is not allowed, i.e., DFA cannot change state without any input character.

For example, below DFA with Σ = {0, 1} accepts all strings ending with 0.

One important thing to note is, **there can be many possible DFAs for a pattern**. A DFA with a minimum number of states is generally preferred.

2) Nondeterministic Finite Automata(NFA):

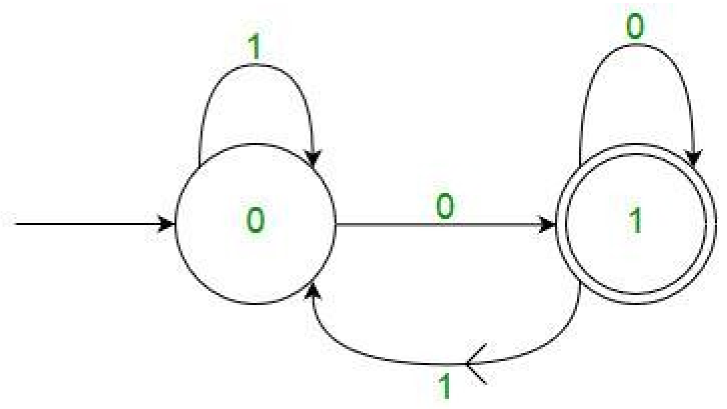
NFA is similar to DFA except following additional features:

Null (or ε) move is allowed i.e., it can move forward without reading symbols.

Ability to transmit to any number of states for a particular input.

However, these above features don't add any power to NFA. If we compare both in terms of power, both are equivalent.

Due to the above additional features, NFA has a different transition function, the rest is the same as DFA.

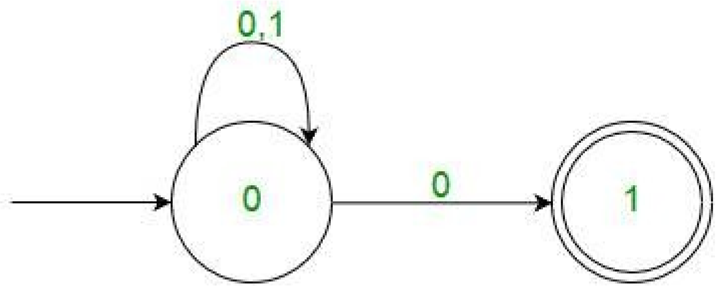


One important thing to note is, there can be many possible DFAs for a pattern. A DFA with a minimum number of states is generally preferred.

δ: Transition Function

δ: Q X (Σ U ε) --> 2 ^ Q.

As you can see in the transition function is for any input including null (or ε), NFA can go to any state number of states. For example, below is an NFA for the above problem.



One important thing to note is, in NFA, if any path for an input string leads to a final state, then the input string is accepted. For example, in the above NFA, there are multiple paths for the input string "00". Since one of the paths leads to a final state, "00" is accepted by the above NFA.

Some Important Points:

Justification:

Since all the tuples in DFA and NFA are the same except for one of the tuples, which is Transition Function (δ)

In case of DFA

δ : Q X Σ --> Q

Now if you observe you'll find out Q X Σ –> Q is part of Q X Σ –> 2Q.

On the RHS side, Q is the subset of 2Q which indicates Q is contained in 2Q or Q is a part of 2Q, however, the reverse isn't true. So mathematically, we can conclude that every DFA is NFA but not vice-versa. Yet there is a way to convert an NFA to DFA, so there exists an equivalent DFA for every NFA.

Both NFA and DFA have the same power and each NFA can be translated into a DFA.

There can be multiple final states in both DFA and NFA.

NFA is more of a theoretical concept.

DFA is used in Lexical Analysis in Compiler.

If the number of states in the NFA is N then, its DFA can have maximum 2N number of states.

Moore Machines: Moore machines are finite state machines with output value and its output depends only on present state. It can be defined as (Q, q0, ∑, O, δ, λ) where:

Q is finite set of states.

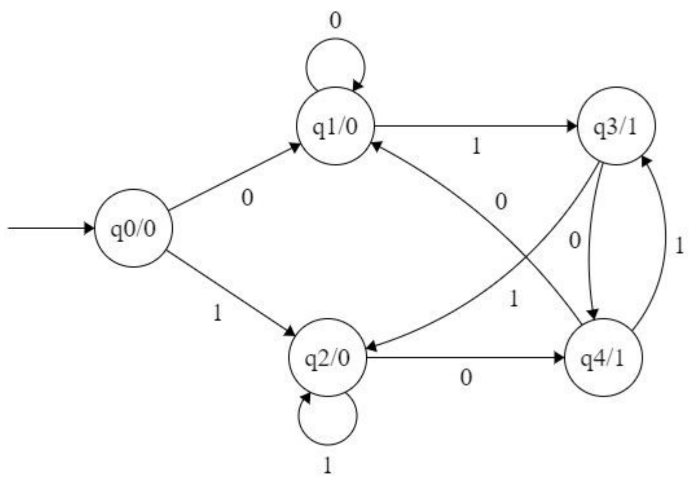
q0 is the initial state.

∑ is the input alphabet.

O is the output alphabet.

δ is transition function which maps Q×∑ → Q.

λ is the output function which maps Q → O.



In the moore machine shown in Figure 1, the output is represented with each input sta te separated by /. The length of output for a moore machine is greater than input by 1.

Input: 11

Transition: δ (q0,11)=> δ(q2,1)=>q2

Output: 000 (0 for q0, 0 for q2 and again 0 for q2)

Mealy Machines: Mealy machines are also finite state machines with output value and its output depends on
present state and current input symbol. It can be defined as (Q, q0, ∑, O, δ, λ') where:

Q is finite set of states.

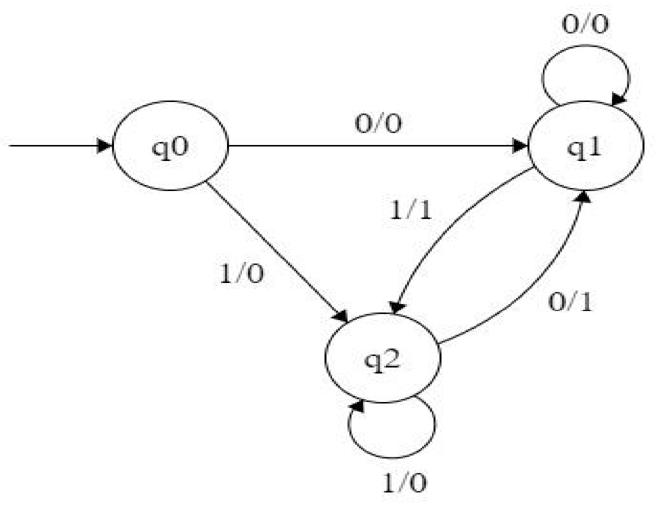
q0 is the initial state.

∑ is the input alphabet.

O is the output alphabet.

δ is transition function which maps Q×∑ → Q.

'λ' is the output function which maps Q×∑→ O.



In the mealy machine shown in Figure 1, the output is represented with each input sy mbol for each state separated
by /. The length of output for a mealy machine is equal to the length of input.

Input:1 1

Transition: δ (q0,11)=> δ(q2,1)=>q2

Output: 00 (q0 to q2 transition has Output 0 and q2 to q2 transition also has Output 0)

NOTE :

If there are n inputs in the Mealy machine then it generates n outputs while if there are n inputs in the Moore machine then it generates n + 1 outputs.

Conversion from Mealy to Moore Machine

Let us take the transition table of mealy machine shown in Figure 2.

	Input=0		Input=1	
Present State	Next State	Output	Next State	Output
q0	q1	0	q2	0
q1	q1	0	q2	1
q2	q1	1	q2	0

Table 1

Step 1. First find out those states which have more than 1 outputs associated with them. q1 and q2 are the states which have both output 0 and 1 associated with them.

Step 2. Create two states for these states. For q1, two states will be q10 (state with output 0) and q11 (state with
output 1). Similarly for q2, two states will be q20 and q21.

Step 3. Create an empty moore machine with new generated state. For moore machine, Output will be associated to each state irrespective of inputs.

Step 4. Fill the entries of next state using mealy machine transition table shown in Table 1. For q0 on input 0,next
state is q10 (q1 with output 0). Similarly, for q0 on input 1, next state is q20 (q2 with output 0). For q1 (both q10 and q11) on input 0, next state is q10. Similarly, for q1(both q10 and q11), next state is q21. For q10, output will be 0 and for q11, output will be 1

	Input=0	Input=1	
Present State	Next State	Next State	Output
q0			
q10			
q11			
q20			
q21			

Table 2

. Similarly, other entries can be filled.

	Input=0	Input=1	
Present State	Next State	Next State	Output
q0	q10	q20	0
q10	q10	q21	0
q11	q10	q21	1
q20	q11	q20	0

q21	q11	q20	1

Table 3

This is the transition table of moore machine shown in Figure 1.

Conversion from moore machine to mealy machine

Let us take the moore machine of Figure 1 and its transition table is shown in Table 3.
Step 1. Construct an empty mealy machine using all states of moore machine as show n in Table 4.

	Input=0		Input=1	
Present State	Next State	Output	Next State	Output
q0				
q10				
q11				
q20				
q21				

Table 4

Step 2: Next state for each state can also be directly found from moore machine transi tion Table as:

	Input=0		Input=1	
Present State	Next State	Output	Next State	Output
q0	q10		q20	
q10	q10		q21	
q11	q10		q21	

q20	q11		q20	
q21	q11		q20	

Table 5

Step 3: As we can see output corresponding to each
input in moore machine transition table. Use this to fill the Output entries.

e.g.; Output corresponding to q10, q11, q20 and q21 are 0, 1, 0 and 1 respectively.

	Input=0		Input=1	
Present State	Next State	Output	Next State	Output
q0	q10	0	q20	0
q10	q10	0	q21	1
q11	q10	0	q21	1
q20	q11	1	q20	0
q21	q11	1	q20	0

Table 6

Step 4: As we can see from table 6, q10 and q11 are similar to each other (same val
ue of next state and Output for
different Input). Similarly, q20 and q21 are also similar. So, q11 and q21 can be elimin
ated.

	Input=0		Input=1	
Present State	Next State	Output	Next State	Output
q0	q10	0	q20	0
q10	q10	0	q21	1

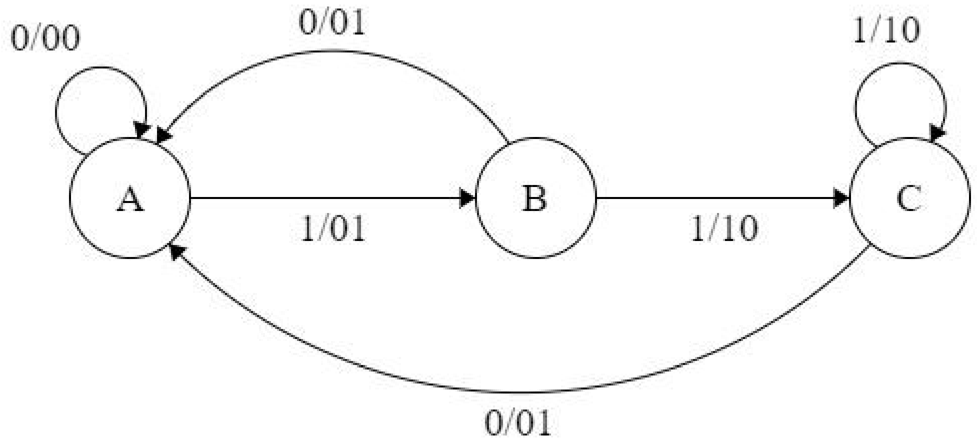
q20	q11	1	q20	0

Table 7

This is the same mealy machine shown in Table 1. So we have converted mealy to mo ore machine and converted back moore to mealy.

Note: Number of states in mealy machine can’t be greater than number of states in moore machine.

Example: The Finite state machine described by the following state diagram with A as starting state, where an arc label is x / y and x stands for 1-bit input and y stands for 2- bit output?



Outputs the sum of the present and the previous bits of the input.

Outputs 01 whenever the input sequence contains 11.

Outputs 00 whenever the input sequence contains 10.

None of these.

Solution: Let us take different inputs and its output and check which option works:

Input: 01

Output: 00 01 (For 0, Output is 00 and state is A. Then, for 1, Output is 01 and state will be B)

Input: 11

Output: 01 10 (For 1, Output is 01 and state is B. Then, for 1, Output is 10 and state is C)

As we can see, it is giving the binary sum of present and previous bit. For first bit, prev ious bit is taken as 0.

This article is contributed by Sonal Tuteja. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Conversion of Epsilon-NFA to NFA

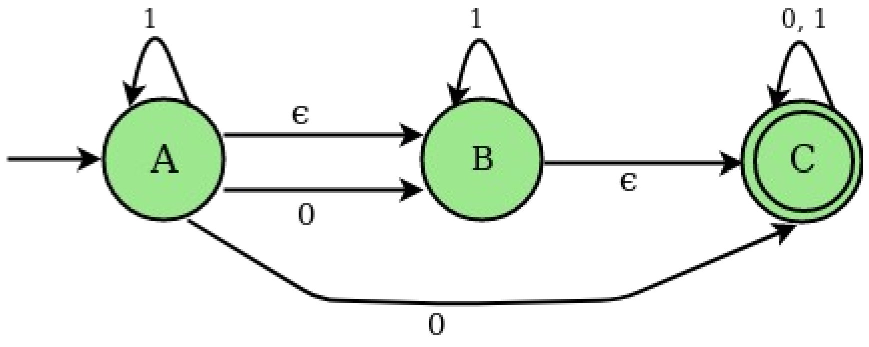
Non-deterministic Finite Automata (NFA) : NFA is a finite automaton where for some cases when a single input is given to a single state, the machine goes to more than 1 states, i.e. some of the moves cannot be uniquely determined by the present state and the present input symbol.

An NFA can be represented as **M = { Q, ∑, ∂, q0, F}**

Q → Finite non-empty set of states. ∑ → Finite non-empty set of input symbols. ∂ → Transitional Function. q0 → Beginning state. F → Final State

NFA with (null) or ∈ move : If any finite automata contains ε (null) move or transaction, then that finite automata is called NFA with ∈ moves

Example : Consider the following figure of NFA with ∈ move :



Transition state table for the above NFA

STATES	0	1	epsilon
A	B, C	A	B
B	–	B	C
C	C	C	–

Epsilon – closure : Epsilon closure for a given state X is a set of states which can be reached from the states X with only (null) or ε moves including the state X itself. In other words, ε-closure for a state can be obtained by union operation of the ε-closure of the states which can be reached from X with a single ε move in recursive manner. For the above example ∈ closure are as follows :

Epsilon closure(A) : {A, B, C}

Epsilon closure(B) : {B, C}

Epsilon closure(C) : {C}

Deterministic Finite Automata (DFA) : DFA is a finite automata where, for all cases, when a single input is given to a single state, the machine goes to a single state, i.e., all the moves of the machine can be uniquely determined by the present state and the present input symbol.

Steps to Convert NFA with ε-move to DFA :

Step 1 : Take ∈ closure for the beginning state of NFA as beginning state of DFA.

Step 2 : Find the states that can be traversed from the present for each input symbol (union of transition value and their closures for each states of NFA present in current state of DFA).

Step 3 : If any new state is found take it as current state and repeat step 2.

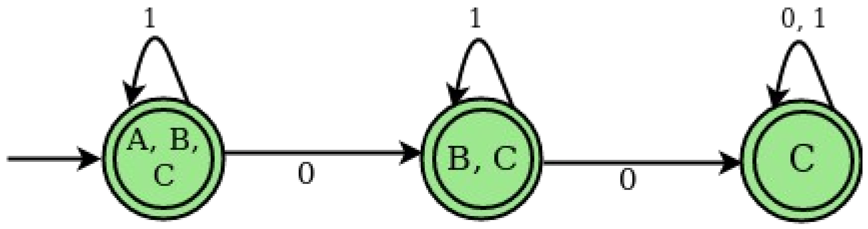
Step 4 : Do repeat Step 2 and Step 3 until no new state present in DFA transition table.

Step 5 : Mark the states of DFA which contains final state of NFA as final states of DFA.

Transition State Table for DFA corresponding to above NFA

STATES	0	1
A, B, C	B, C	A, B, C
B, C	C	B, C
C	C	C

DFA STATE DIAGRAM



Examples :

Input : 6 2

```
FC - BF
- C -
- - D
E A -
A - BF
- - -
```

Output :

```
 STATES OF NFA :        A, B, C, D, E, F,

 GIVEN SYMBOLS FOR NFA:     0, 1, eps
 NFA STATE TRANSITION TABLE
STATES        |0      |1        eps
--------+------------------------------------
A             |FC     |-        |BF
B             |-      |C        |-
C             |-      |-        |D
D             |E      |A        |-
E             |A      |-        |BF
F             |-      |-        |-

 e-Closure (A) :    ABF
 e-Closure (B) :    B
 e-Closure (C) :    CD
 e-Closure (D) :    D
 e-Closure (E) :    BEF
 e-Closure (F) :    F
*****************************************************
        DFA TRANSITION STATE TABLE
 STATES OF DFA :        ABF, CDF, CD, BEF,
 GIVEN SYMBOLS FOR DFA:     0, 1,
STATES    |0    |1
--------+-----------------------
ABF    |CDF     |CD
CDF    |BEF     |ABF
CD    |BEF     |ABF
BEF    |ABF     |CD
```

Input :9 2- - BH- - CE D - - - G- F -- - G- - BHI - -- - -

Output :

```
STATES OF NFA :        A, B, C, D, E, F, G, H, I,
GIVEN SYMBOLS FOR NFA:     0, 1, eps
NFA STATE TRANSITION TABLE
STATES         |0    |1    eps
--------+-----------------------------------
A                    |-     |-     |BH
B                     |-     |-     |CE
C                   |D     |-     |-
D                   |-     |-     |G
E                   |-     |F     |-
F                   |-     |-     |G
G                   |-     |-     |BH
H                    |I     |-     |-
I                    |-     |-     |-
```

```
 e-Closure (A) :    ABCEH

 e-Closure (B) :    BCE

 e-Closure (C) :    C

 e-Closure (D) :    BCDEGH

 e-Closure (E) :    E

 e-Closure (F) :    BCEFGH

 e-Closure (G) :    BCEGH

 e-Closure (H) :    H

 e-Closure (I) :    I

*****************************************************
        DFA TRANSITION STATE TABLE
 STATES OF DFA :        ABCEH, BCDEGHI, BCEFGH,
 GIVEN SYMBOLS FOR DFA:     0, 1,
STATES    |0    |1
--------+----------------------
ABCEH    |BCDEGHI     |BCEFGH
BCDEGHI    |BCDEGHI     |BCEFGH
BCEFGH    |BCDEGHI     |BCEFGH
```