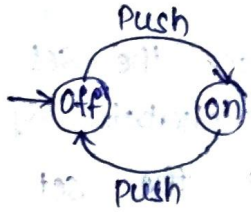
FLAT

1. Why study the Automata Theory?

Finite Automata are useful model for many important kind of Hardware and software.

* software for designing and checking the behavior of digital circuits.
* The lexical Analyser of a typical compiler (it breaks the input text into logical units like identifiers, keywords and so on.
* software for very verify systems of all types that have finite number of distinct states, such as protocols for secure exchange of information.

Example:1: The simplest non trivial finite Automata is an on off switch. It has two states on, off states. It allows the user to press the switch to change the state of switch from on to off (or) off to on.



Here on and off are states and arcs between states are labelled by inputs like push.

* Structural Representations:

There are two important notations that are not automata, but plays an important role in the study of Automata and their Applications.

1. Grammars: Grammars are useful models when defining softw-are that process data with a recursive structure. The best known example is parser.

* $E = E+E/E-E/E*E/E$

2. Regular Expressions: Regular Expressions also denote the structur-e of data especially "text, strings"

* Central concepts of Automata theory:

1) Alphabet:

An Alphabet is a finite, non empty set of symbols. We use the symbol Σ for an alphabet. common Alphabets include

ex:- $\Sigma = \{0, 1\}$ ⇒ Binary Alphabet

$\Sigma = \{a, b, c, \dots, z\}$ ⇒ set of all the alphabets (lower case letters).

2) Strings:

A string is a finite sequence of symbols choosen from some alphabets.

Ex:- 0 1 1 0 1 is a string from binary alphabet

$\Sigma = \{0, 1\}$

3) Empty string:

The empty string is the string with zero occurances of symbols. It is denoted by ϵ or ε.

4) Length of a string:

It is useful to classify strings by their lengths i.e., the number of positions for symbol in the string.

if w is a string then the length of the string is denoted by $|w|$.

Ex:- If $w = 0\,1\,1\,0\,1$ Then $|w| = 5$

if $w = 0\,1\,\varepsilon$ Then $|w| = 0$

5) Powers of Alphabet:

If Σ is an Alphabet, we can express the set of all strings of certain length from that alphabet by using exponential notation. we define Σ^k to be the set of strings of length k, each of those symbols is in Σ.

Ex:- $\Sigma = \{a, b, c\}$ then $\Sigma^1 = \{a, b, c\}$

$\Sigma^0 = \varepsilon$; $\Sigma^2 = \{ab, bc, ca \text{----}\}$

Set of all strings over an alphabet Σ is denoted by Σ^*.

Ex:- $\Sigma = \{0, 1\}$ then $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000 \text{----}\}$

$\Sigma^* = \{\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \text{---}\}$

The set of non empty strings from alphabet Σ is denoted by Σ^+.

* Concatenation of strings:

*	$\Rightarrow$	0	to ∞
+	$\Rightarrow$	1	to ∞

let x & y be the strings then xy denotes the concatenation of x & y i.e., the string formed by making a copy of x & followed it by copy of y.

Ex:- let $x = 1101$ & $y = 0011$

then $xy = 1101\,0011$ $yx = 0011\,1101$

* languages:

set of strings of all which are choosen from Σ^*, where Σ is an alphabet, is called a language.

If Σ is an alphabet & $L \subseteq \Sigma^*$ then L is a language over Σ.

Ex:- ① The language of all strings consisting of n zeroes (0's) followed by n 1's for some $n \geq 0$.

$\{\varepsilon, 01, 0011, 000111\}$ -----

② set of strings of 0's and 1's with equal number of strings.

$\{\varepsilon, 0, 1, 1, 0, 0011, 0101, 1001, \text{-----}\}$

③ Σ^* is a language for any alphabet Σ.

④ ϕ The empty language is a language over any alphabet.

$S \rightarrow aB \mid bA$

$A \rightarrow a \mid as \mid bAA$

$B \rightarrow b \mid bs \mid aBB$

LMD

S
aB
aBB B
aaa BBB
aaab BB
aaabbsB
aaabbaBB
aaabbabB
aaabbabb

* RMD for the string 0100110

$S \rightarrow 0S \mid 1AA$

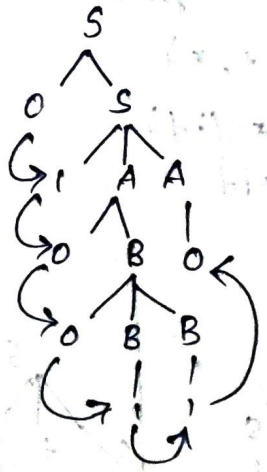
$A \rightarrow 0 \mid 1A \mid 0B$

$B \rightarrow 1 \mid 0BB$

sol:-

S
0 S
0 1 A A
0 1 A 0
0 1 0 B 0
0 1 0 0 B B 0
0 1 0 0 B 1 0
0 1 0 0 1 1 0

Parse tree:



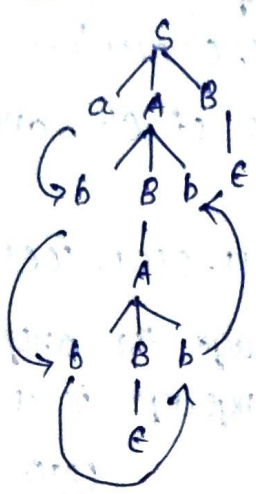
* $S \rightarrow aAB$
  $A \rightarrow bBb$
  $B \rightarrow A/\epsilon$
  string abbbb

$V = \{S, A, B\}$ Non terminals
$T = \{a, b, \epsilon\}$ Terminals
ϵ = Epsula (null string)

LMD:
Left Most Derivation

S
a A B
a b B b B
a b A b B
a b b B b b B
a b b ϵ b b B
a b b b B
a b b b b ϵ
a b b b b

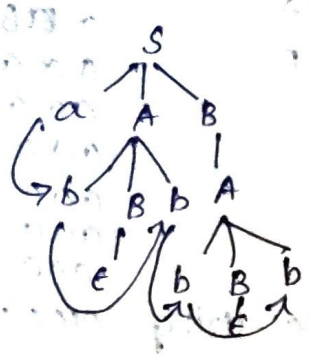
Parse Tree:



RMD:
Right Most Derivation

S
a A B
a A A
a A b B b
a A b ϵ b
a A b b
a b B b b
a b ϵ b b b
a b b b b

Parse Tree:



*Ambigous grammer:

A grammer is said to be ambigous if we can generate more than one parse tree for the same string.

Ex:- prove that the grammer is Ambigous.

$V = \{S\}$ $\Sigma = \{id, +, *\}$

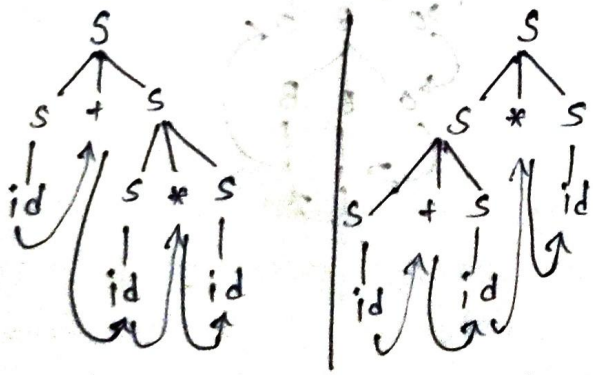
$\Sigma \Rightarrow$ Sigma is notation for set of terminals.

$P = \{S \rightarrow S+S/S*S/id\}$

$P \Rightarrow$ set of productions.

String id+id*id

LMD:



We construct two parse trees it is Ambigous.

* check whether the grammer is Ambigous or not

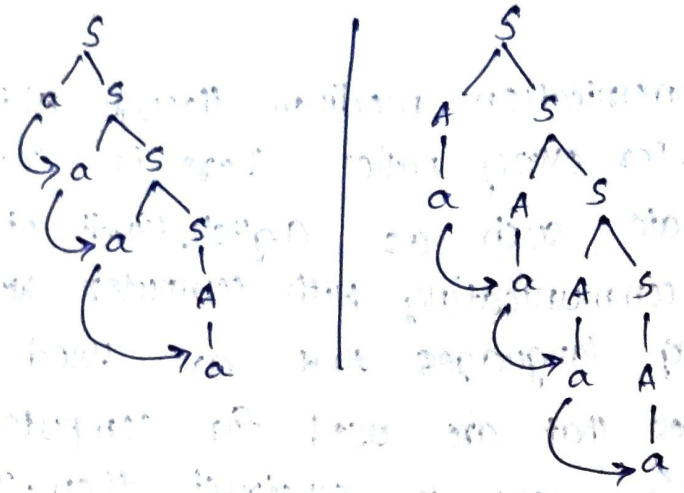
$S \rightarrow aS/As/A$

$A \rightarrow As/a$ String aaaa

$V = \{S, A\}$ Non terminals

$T = \{a\}$ Terminals

LMD:



It is Ambigous.

*Simplification of CFG: context Free Grammer

CFG

$S \rightarrow A$
$A \rightarrow B$
$B \rightarrow C$
$C \rightarrow D$
$D \rightarrow a/A$

S
A
B
C
D
a

$S \rightarrow AB/a$
$A \rightarrow a$
B has no derivation.

* $S \rightarrow a$

S
a

context Free Grammer can be simplified by:

*useless production Elimination
*Unit production elimination
*ε production elimination.

*Useless production Elimination:

$S \rightarrow AB/a$
$A \rightarrow a$

LMD

S
A B
a (B) useless

After removing useless symbol

$S \rightarrow a$
$A \rightarrow a.$

S → aSa | Abd | Bde

A → Ada

B → BBB | a

* Formal languages:

Language is a communication medium through which two persons communicate. For every nation there is some language to communicate such as English, Hindi etc. In the same manner for communicating with computer we are using several programming languages that are used for like C, C++, java. languages that are used for computer programming have some grammers to construct them. These Grammers and rules are used for checking and converting data into machine understandable format.

* Automata theory:

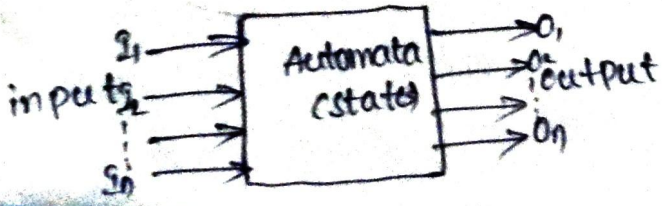
Automata theory is the study of abstract machines and computational problems related to those abstract machi-nes.

Ex: In computer all processes are appeared to be automa-tically. A given input is processed in the CPU and output is generated. We are not concerend about internal operations in the CPU. These types of machine should be called as a Abstract machines.

These Abstract machines are also called Automata. The word Automata is from a greek word which means that something is doing something by itself.

* Definition of Automata:

An Automata is a system where materials, energy or information are transformed and transmitted for performing some operations without direct participation of human.



* Characteristics:

*Input: $I_1, I_2 \dots I_n$ are number of Input lines, n number of inputs will be taken in each single time instance. Input of each input line is finite and taken from a set of Alphabet (Σ)

*Output: $O_1, O_2 \dots O_n$ are outputs, that are generated from each output line is finite and these are belongs to Alphabet (Σ).

*state: At any time the Automata can be in one of the states $Q_0, Q_1, Q_2 \dots Q_n$. The state belongs to a set called state 'Q'.

*State Transition: The state in which the automata designed by getting a particular input is determined by state transition. The state Transition is a function of present state and input, which produces the next state.

*Output Relation: Similar to the state transition for state, there is a relation for output. The output depends either on present state and present input (or) present state only.

*Uses of Automata:

- The first phase of compiler called lexical Analyser is designed by finite Automata. In syntax analysis's part the errors are checked by the rules of context Free Grammer.

*Working principal of software used for checking the Behaviour of Digital circuits is based on Automata Theory.

*Automata is useful for designing the software for counting the occurance of particular goal, pattern and so on in large text.

*Finite Automata:

- It is one type of finite state machine. It has finite number of states. It can be the thought of with a finite state machine without outputs. Automata can be described as an input tape containing the input symbols (Σ) with a reading head scanning the input from left to right. The inputs are fed into finite control which contains the transactional functions (δ). According to the transactional function state 'Q' changes occur.

Input tape contains input symbols. It is divided into several squares which contains single character of input Alphabet. Each ends of input tape contains end marker ($).

* Reading head:

The head scans each square in the input tape and reads input from the tape. It moves from mostly left to right.

* Finite control:

It can be considered as control unit of finite Automata. The Reading head scans the input tape and send it to the finite control. This finite control decide that the machine is in the state and it gets this input, so it will go to this state. All state transition Relations are written in this finite control.

* Finite Automata:

* Finite Automata are the machine formats of Regular expression which is the longest format of Type-III grammar. Finite Automata is defined as $M = \{Q, \Sigma, \delta, q_0, F\}$

Five Tuple Representation of Finite Automata.

Where Q = Finite non-empty set of states.

Σ = Finite non-empty set of input symbol.

δ = Transitional function

q_0 = Beginning state

F = Finite non-empty set of Final state.

Transitional System:

Transitional system is Finite Directed graph in which each node represent a state and directed arc indicates transition of the state.

Label of arc indicates input or output or both.

Transitional function has two properties.

1. $\delta(q, \Lambda) = q$. It means if input is given null for state machine remains in the same state.

2. For all strings x and input symbol A belongs to Σ then $\delta(q, ax) = \delta(\delta(q,a), x)$

$\delta(q, xa) = \delta(\delta(q,x), a)$

*Graphical Representation of Finite Automata:

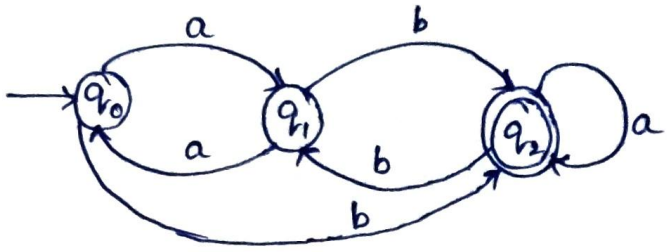
In Graphical Representation,

State is Represented as (q)

Beginning state is Represented as → (q)

Final State is Represented as ((q))

Example:



Where

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$\delta(q_0, a) = q_1$

$\delta(q_0, b) = q_2$

$\delta(q_1, a) = q_0$

$\delta(q_1, b) = q_2$

$\delta(q_2, a) = q_2$

$\delta(q_2, b) = q_1$

$q_0 = \{q_0\}$

$F = \{q_2\}$

*Tabular Format:

In Tabular Format a state is represented by name of the state.

Beginning State is Represented as → q

Final State is Represented as (q)

Tabular Format of above Finite Automata is

Previous state	next state. a	b
→ q_0	q_1	q_2
q_1	q_0	q_2
(q_2)	q_2	q_1

where $Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$

$\delta(q_0, a) = q_1$

$\delta(q_0, b) = q_2$

$\delta(q_1, a) = q_0$

$\delta(q_1, b) = q_2$

$\delta(q_2, a) = q_2$

$\delta(q_2, b) = q_1$

$q_0 = \{q_0\}$

$F = \{q_2\}$

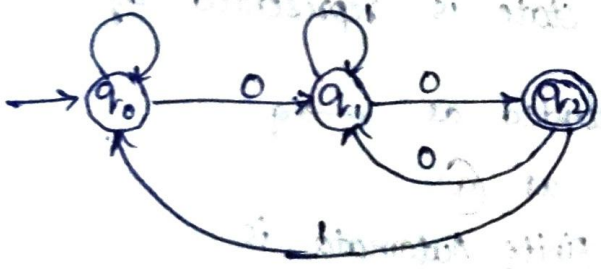
* Acceptance of string by Finite Automata:

There are TWO conditions for declaring string to be accepted by Finite Automata. Those are:

- string must be totally Traversed.
- Machine must come to Final state.

* Test whether the given strings

~~0,1,00,1,0~~ 010010 and ~~0,1,0,1,0~~ 01010 are Accepted by given Finite Automata.



$\delta(q_0, 010010) = \delta(\delta(q_0, 0), 10010)$

$= \delta(q_1, 10010)$

$= \delta(\delta(q_1, 1), 0010)$

$= \delta(q_1, 0010)$

$= \delta(\delta(q_1, 0), 010))$

$= \delta(q_2, 010)$

$= \delta(\delta(q_2, 0), 10)$

$= \delta(q_1, 10)$

$= \delta(\delta(q_1, 1), 0)$

$= \delta(q_1, 0)$

$= q_2$

q_2 is Final state the given string is Accepted by Finite Automata.

$\delta(q_0, 01010) = \delta(\delta(q_0, 0), 1010)$

$= \delta(q_1, 1010)$

$= \delta(\delta(q_1, 1), 010)$

$= \delta(q_1, 010)$

$= \delta(\delta(q_1, 0), 10)$

$= \delta(q_2, 10)$

$= \delta(\delta(q_2, 1), 0)$

$= \delta(q_0, 0)$

$= q_1$

the string is not Accepted by given Finite Automata

* 011001

01010

state	Input 0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_3
q_3	q_1	q_3

$\delta(q_0, 01010) = \delta(\delta(q_0, 0), 1010)$

$= \delta(q_0, 1010)$

$= \delta(\delta(q_0, 1), 010)$

$= \delta(q_1, 010)$

$= \delta(\delta(q_1, 0), 10)$

$= \delta(q_2, 10)$

$= \delta(\delta(q_2, 1), 0)$

$= \delta(q_3, 0)$

$= q_1$

Not Accepted.

$\delta(q_0, 01100) = \delta(\delta(q_0, 0), 1100)$

$= \delta(q_0, 1100)$

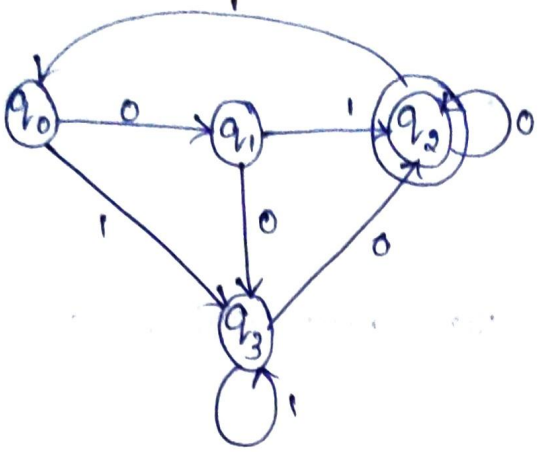
$= \delta(\delta(q_0, 1), 100)$

$= \delta(q_1, 100)$

$= \delta(\delta(q_1, 1), 00)$

$= \delta(q_2, 00)$

$= \delta(\delta(q_2, 0), 0) = \delta(q_0, 0) = \delta(\delta(q_1, 0), 1) = \delta(q_0, 1) = q_3$



0111100
111111
11010

$\delta(q_0, 0111100) = \delta(\delta(q_0,0), 111100)$

$= \delta(q_1, 111100)$

$= \delta(\delta(q_1,1), 11100)$

$= \delta(q_2, 11100)$

$= \delta(\delta(q_2,1), 1100)$

$= \delta(q_0, 1100)$

$= \delta(\delta(q_0,1), 1100)$

$= \delta(q_3, 1100)$

$= \delta(\delta(q_3,1), 100)$

$= \delta(q_3, 100)$

$= \delta(\delta(q_3,1), 00)$

$= \delta(q_3, 00)$

$= \delta(\delta(q_3,0), 0)$

$= \delta(q_2, 0)$

$= \delta(\delta(q_2,0)) = q_2.$

Accepted by Finite Automata.

$\delta(q_0, 111111) = \delta(\delta(q_0,1), 11111)$

$= \delta(q_3, 11111)$

$= \delta(\delta(q_3,1), 1111)$

$= \delta(q_3, 1111)$

$= \delta(\delta(q_3,1), 111)$

$= \delta(q_3, 111)$

$= \delta(\delta(q_3,1), 11)$

$= \delta(q_3, 11)$

$= \delta(\delta(q_3,1), 1)$

$= \delta(q_3, 1)$

$= q_3$

Not Accepted by Finite Automata.

$\delta(q_0, 11010) = \delta(\delta(q_0, 1), 1010)$

$= \delta(q_3, 1010)$

$= \delta(\delta(q_3, 1), 010)$

$= \delta(q_3, 010)$

$= \delta(\delta(q_3, 0), 10)$

$= \delta(q_2, 10)$

$= \delta(\delta(q_2, 1), 0)$

$= \delta(q_0, 0)$

$= q_1$

Not Accepted by Finite Automata.

*

States	Input 0	Input 1
q_0	q_2	q_3
q_1	q_0	q_2
q_2	q_1	q_3
q_3	q_3	q_1

0001101
00000
01010

$\delta(q_0, 0001101) = \delta(\delta(q_0, 0), 001101)$

$= \delta(q_2, 001101)$

$= \delta(\delta(q_2, 0), 01101)$

$= \delta(q_1, 01101)$

$= \delta(\delta(q_1, 0), 1101)$

$= \delta(q_0, 1101)$

$= \delta(\delta(q_0, 1), 101)$

$= \delta(q_3, 101)$

$= \delta(\delta(q_3, 1), 01)$

$= \delta(q_1, 01)$

$= \delta(\delta(q_1, 0), 1)$

$= \delta(q_0, 1)$

$= q_3$

Accepted by Finite Acctomata.

$\delta(q_0, 00000) = \delta(\delta(q_0, 0), 0000)$

$= \delta(q_2, 0000)$

$= \delta(\delta(q_2, 0), 000)$

$= \delta(q_1, 000)$

$= \delta(\delta(q_1, 0), 00)$

$= \delta(q_0, 00)$

$= \delta(\delta(q_0, 0), 0)$

$= \delta(q_2, 0)$

$= q_1$

Not Accepted by Fhite Acctomata.

$\delta(q_0, 01010) = \delta(\delta(q_0, 0), 1010)$

$= \delta(q_2, 1010)$

$= \delta(\delta(q_2, 1), 010)$

$= \delta(q_3, 010)$

$= \delta(\delta(q_3, 0), 10)$

$= \delta(q_3, 10)$

$= \delta(\delta(q_3, 1), 0)$

$= \delta(q_1, 0)$

$= q_0$

Not Accepted by Finite Automata.

* Types of Finite Automata:

There are two types of Finite Automata.

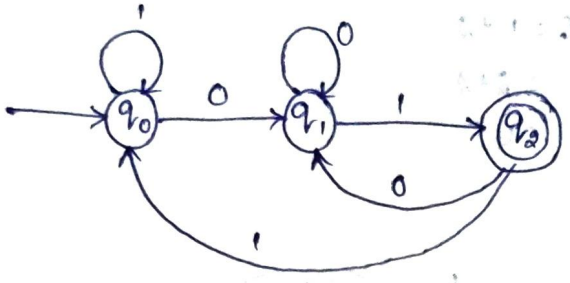
1. Deterministic Finite Automata (DFA)
2. NonDeterministic Finite Automata (NFA)

1. Deterministic Finite Automata:

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

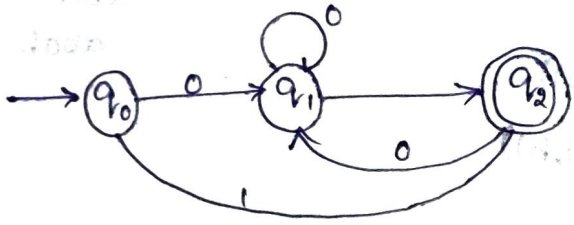
DFA can be represented as $M_{DFA} = \{Q, \Sigma, \delta, q_0, F\}$

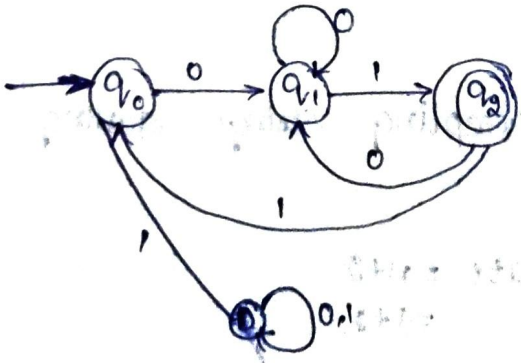
where, δ is a transitional function mapping $Q \times \Sigma = Q$ where $|Q| = 1$



Dead state:

It is a state where the control can enter and to be confined till the input ended, but there is no way to come out from the state. (the string is dead) are finished on entering the state.





Steps:

Type 1: First string starting with particular substring.

- Decide the minimum number of states required in DFA. Draw them.

Rule: All strings starting with length n, substring will always

requires minimum n+2 states in DFA.

- Decide strings for which you will construct DFA.
- construct DFA for above decide strings.

Note: Always prefer to go with existing path. Create a new path only when you can't find a new path to go

- After Drawing DFA above decided string, send the left possible combinations to dead state not over starting st

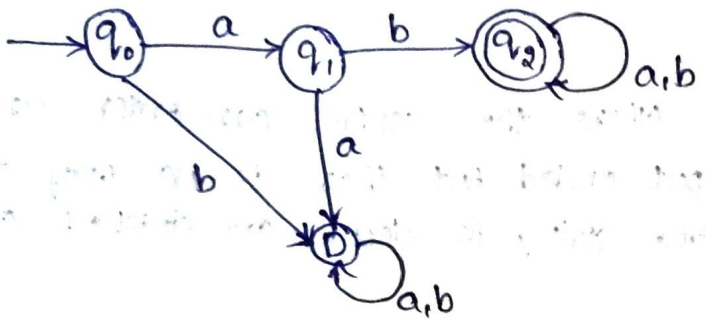
*Draw DFA for language accepting strings starting with ab ov input Alphabet $\Sigma = \{a, b\}$

$|ab| = 2+2$

No.of minimum state $= n+2$

$= 2+2$

$= 4$



ab
aba abb
abaa abbb
abab abba

- $\sigma(\sigma(q_0, ab))$

$= \sigma(\sigma(q_0, a), b)$

$= \sigma(q_1, b)$

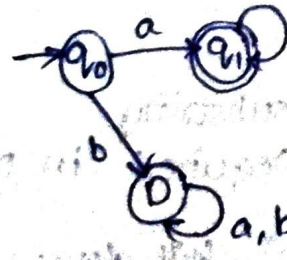
$= q_2$

* Draw DFA for language accepting strings starting with a ove the Alphabet. $\Sigma = \{a, b\}$

Number of Minimum state. $= n+2$

$= 1+2$

$= 3$

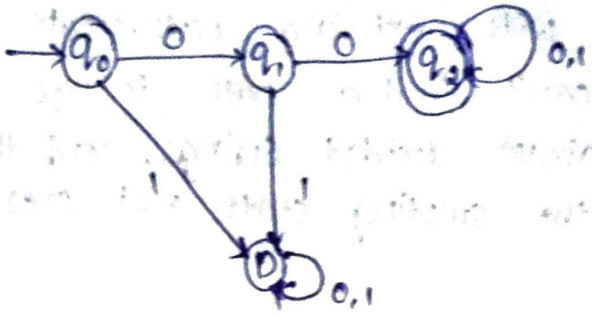


a aaa
aa aba
ab aab
abb

* Draw DFA for language Accepting strings ~~ending~~ starting with 00 Alphabet $\Sigma = \{0,1\}$

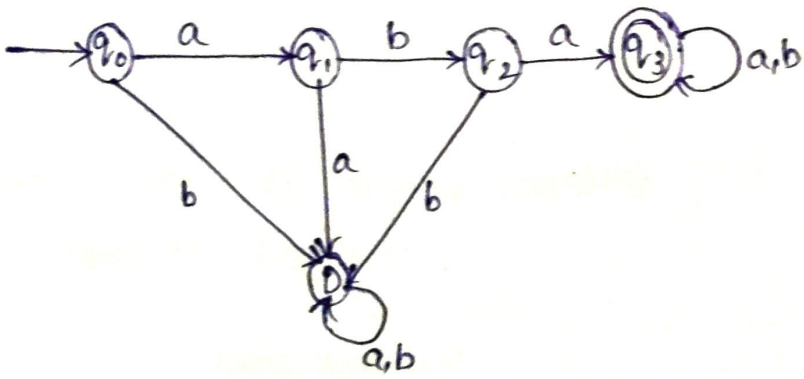
No of minimum state = n+2
= 2+2
= 4

00
000
001
0010
0001
0011



* Draw DFA for language Accepting strings ~~starting~~ with aba Alphabet $\Sigma = \{a,b\}$.

No of minimum state = n+2
= 3+2
= 5



aba abab
abaa abaab
abaaa ababa
~~abab~~ ababb
~~aba~~

Types: For strings ending with particular substrings.

- Decide minimum no.of states requiring DFA. Draw them.

Rule: All strings ending with n length substring will always require minimum n+1 states in DFA.

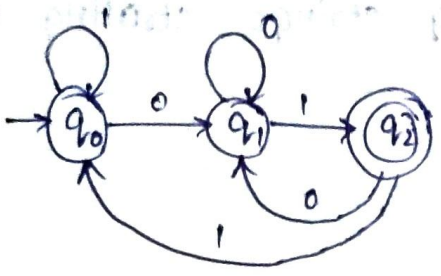
- Decide strings for which you will construct DFA
- construct DFA for above decided strings.

Note: Always prefer to go with existing path. create a new path only when you can't find a path to go.

- After drawing DFA from above decided strings, send the left possible combinations to the starting state not over dead state.

* Draw DFA for language Accepting strings ending with 01 over Alphabet $\Sigma = \{0,1\}$

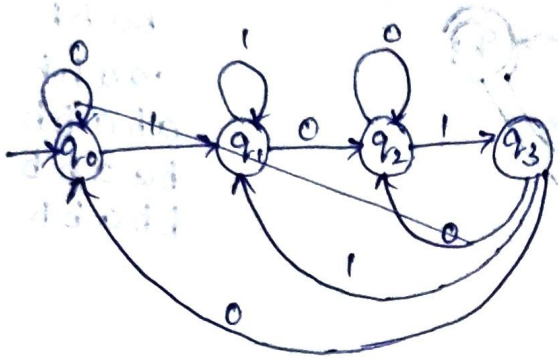
No.of minimum state = n+1
= 2+1
= 3



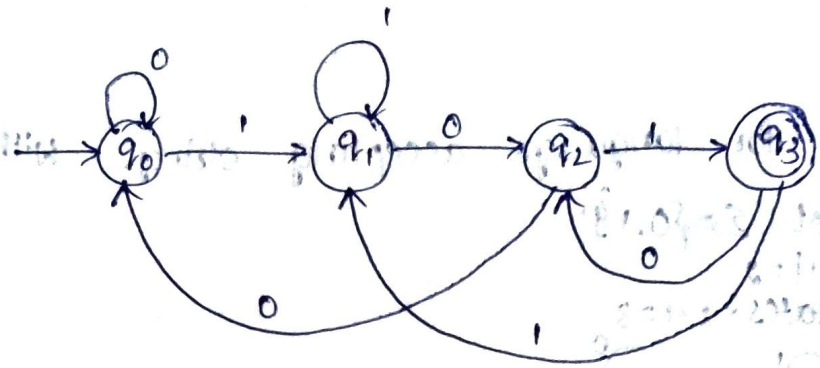
01
001
101
0001
0101
1001
1101

* Draw DFA for language accepting strings ending with 101 over Alphabet $\Sigma = \{0,1\}$.

No of minimum string = 3+1
= 4

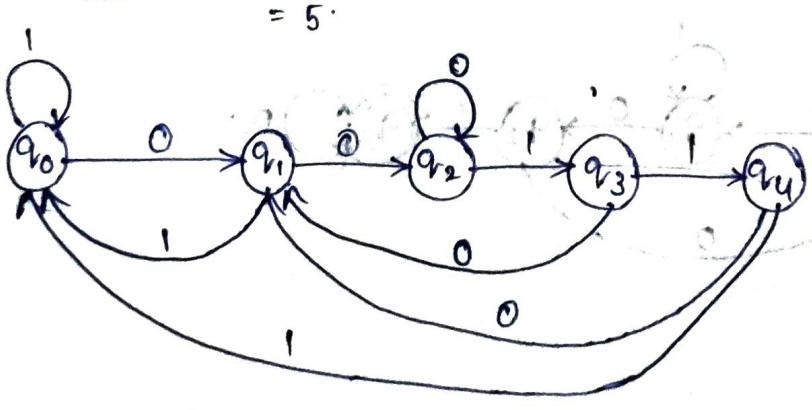


101
0101
1101
00101
01101
10101
11101



* Draw DFA for language Accepting strings ending with 0011 over Alphabet $\Sigma = \{0,1\}$.

States n+1 = 4+1
= 5



0011
00011
10011
000011
010011
110011
100011

* Draw DFA for language accepting strings ending with abb over alphabet $\Sigma = \{a, b\}$.

States = 3+1 = 4

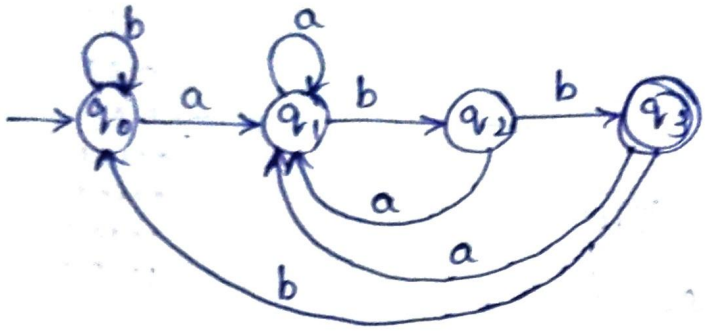
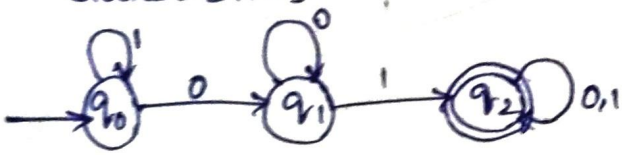


abb
aabb
babb
aaabb
ababb
baabb
bbabb

* Draw DFA for language accepting strings with 01 over Alphabet $\Sigma = \{0, 1\}$.

|01| = 2

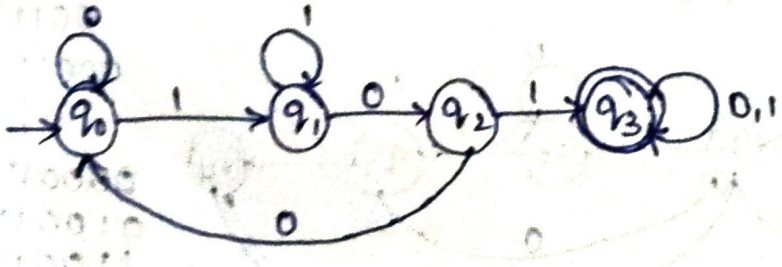
States = 2+1 = 3



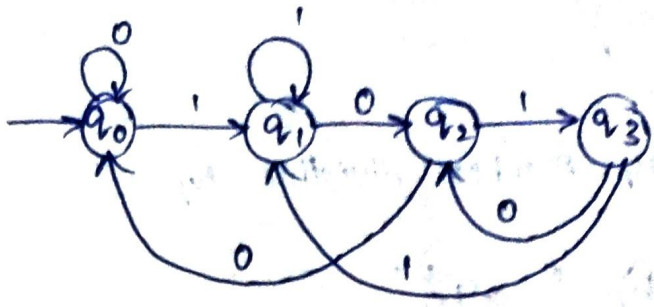
* Draw DFA for language accepting strings with 101 over an Alphabet $\Sigma = \{0, 1\}$

|101| = 3

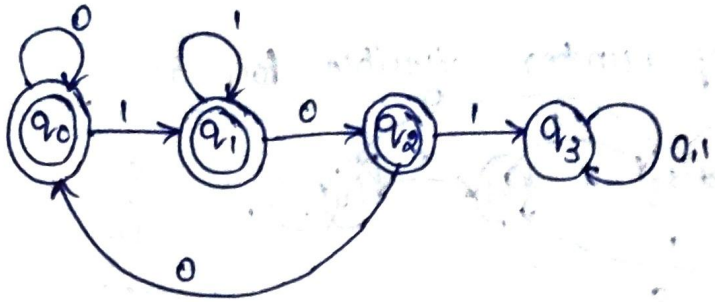
3+1 = 4



* Ending with 101



not Accept string 101.



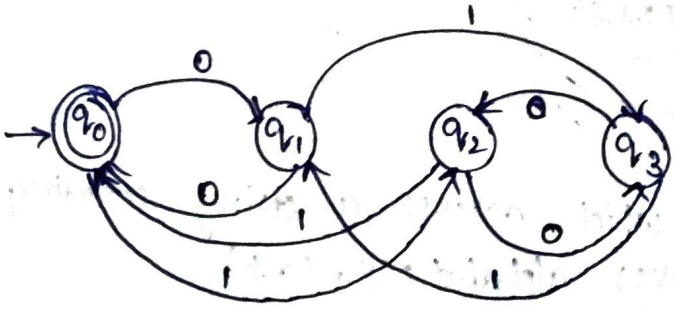
*Draw DFA for a language which Accept strings of even number of 0's and Even number of 1's over Alphabet-{0,1}

q_0 ① even 0's even 1's

q_1 ② odd 0's even 1's

q_2 ③ even 0's odd 1's

q_3 ④ odd 0's odd 1's



Strings Acceptance:

$\delta(\delta(q_0, 0011))$

$\delta(\delta(q_0, 0), 011)$

$\delta(q_1, 011)$

$\delta(\delta(q_1, 0), 11)$

$\delta(q_0, 11)$

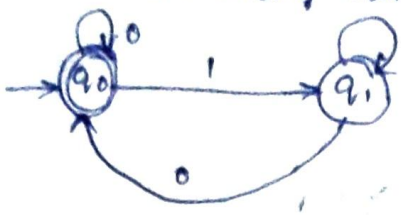
$\delta(\delta(q_0, 1), 1)$

$\delta(q_2, 1)$

q_0

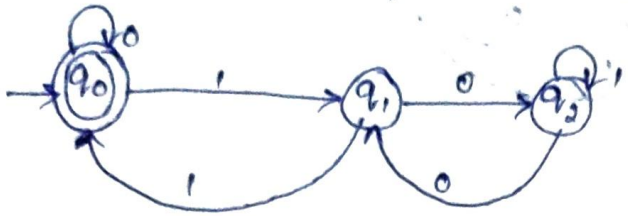
$\therefore q_0$ is the Final state

* Design DFA for binary number divisible by 2

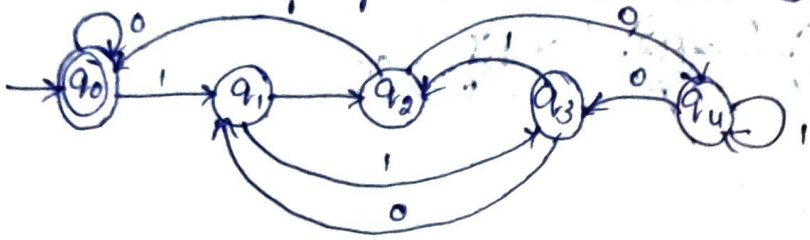


1000

* Design DFA for binary number divisible by 3

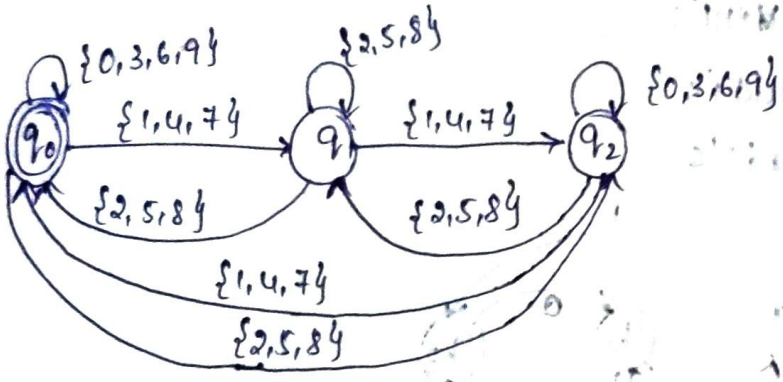


* Design DFA for binary number divisible by 5.

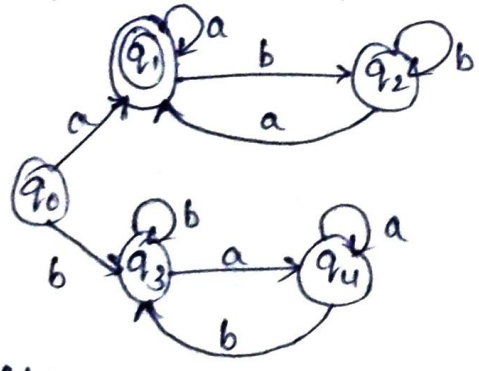


00
01
10
11
100
101
110
111
1000
1001

* Design DFA for decimal number divisible by 3 over alphabet $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$



* Design DFA for language which accepts a string starting & ending with some symbol over alphabet = $\Sigma = \{a,b\}$



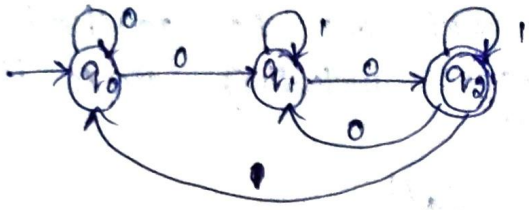
* NFA:

NFA is a finite automata where for some cases when a single input to a single state the machine goes to more than one state i.e., some moves of machine cannot be uniquely determined

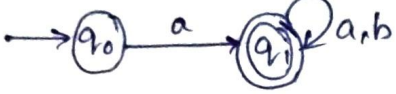
by present state and present input symbol.

$M_{NFA} = \{Q, \Sigma, \delta, q_0, F\}$

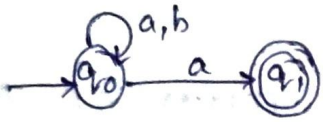
where δ is transitional function mapping $Q \times \Sigma = 2^Q$ where $2Q$ is power set of Q.



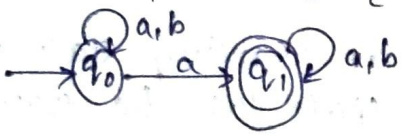
1. Design NFA which accepts a string starting with a over an alphabet $\Sigma = \{a,b\}$



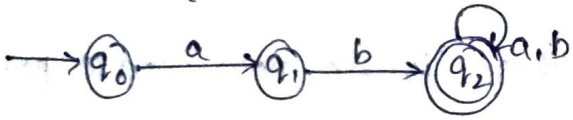
2. Design NFA which accepts a string ending with a. over an alphabet $\Sigma = \{a,b\}$



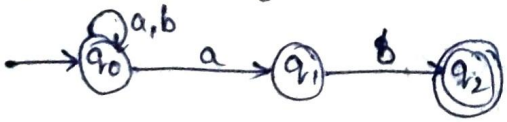
3. Design NFA for a language which accepts a string with a over an Alphabet $\Sigma = \{a,b\}$



4. Design NFA which accepts a string starting with a,b over an alphabet $\Sigma = \{a,b\}$



5. Design NFA which accepts a string ending with ab over an alphabet $\Sigma = \{a,b\}$



6. Design NFA which accepts a string with ab over an alphabet $\Sigma = \{a,b\}$

*NFA with Σ(null):-

If any finite automata contains any Σ(null) moves or transitions then that finite automata for input ϵ the machine goes to more than one state so finite automata with Σ moves is called NFA

NFA with Σ moves can be defined as

$M_{NFA\ null} = \{Q, \Sigma, \delta, q_0, F\}$

Σ is set of alphabets including Σ.
δ is a transitional function mapping with $Q \times \Sigma = 2^Q$
where 2^Q is power set of Q.

*Σ-closure:

In that state set of states which can be reached from given state with only Σ(null) moves including state itself is called Σ-closure of that state.

Σ-closure of $q_0 = \{q_0, q_1, q_2\}$

Σ-closure of $q_1 = \{q_1, q_2\}$

Σ-closure of $q_2 = \{q_2\}$

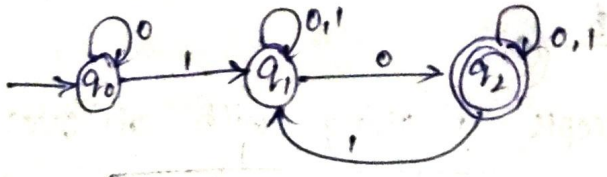
NFA TO DFA CONVERSION:

let $M = \{Q, \Sigma, \delta, q_0, F\}$ is NFA then after conversion final DFA is defined by $M' = \{Q', \Sigma', q_0, \delta', F'\}$

procedure:

1. We will take starting state of NFA as starting state of DFA.
2. Find states for each input symbol that can be traversed from starting state or present state. Draw transition table of DFA
3. If we from new state take it as a current state and repeat step 2.
4. Repeat step2 and step3 until there is no new state present in transition table of DFA
5. Mark states of DFA as final state which contains final state of NFA.

Ex:



	0	1
→ q_0	q_0	q_1
q_1	$\{q_1, q_2\}$	q_2
q_2	ϕ	$\{q_1, q_2\}$

We will obtain δ' transition for state q_0

$\delta'(q_0, 0) = q_0$.

$\delta'(q_0, 1) = q_1$ (new state)

we will obtain δ' transition new state q_1,

$\delta'(q_1,0) = \delta(\{q_1,q_2\},0)$

$= \delta(q_1,0) \cup \delta(q_2,0)$

$= \{q_1,q_2\} \cup \{q_2\}$ new state.

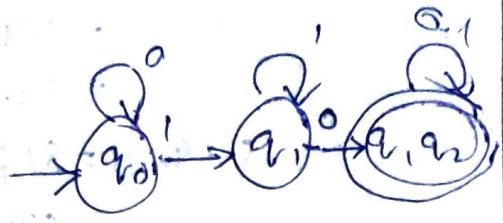
	0	1
q_0	q_0	q_1
q_1	$\{q_1,q_2\}$	q_1
$\{q_1,q_2\}$	$\{q_1,q_2\}$	$\{q_1,q_2\}$

$\delta'(q_1,1) = q_1$

$\delta'(\{q_1,q_2\},0) = \delta(\{q_1,q_2\},0)$

$= \delta(q_1,0) \cup \delta(q_2,1)$

$= \{q_1,q_2\} \cup \{q_2\}$



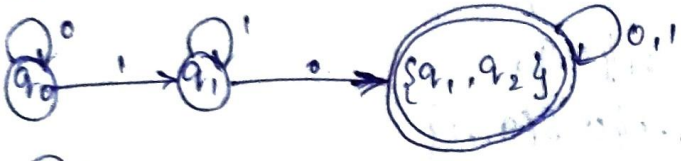
$\delta'(\{q_1,q_2\},1) = \delta(\{q_1,q_2\},1)$.

$= \delta(q_1,1) \cup \delta(q_2,1)$

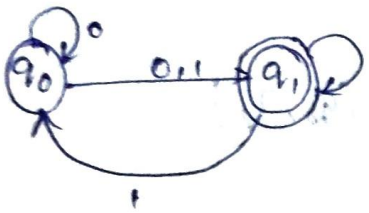
$= \{q_1,1\} \cup \{q_2,1\}$

$= \{q_1\} \cup \{q_1,q_2\} = \{q_1,q_2\}$

Find state DFA:-



(2)



states	input 0	input 1
q_0	$\{q_0,q_1\}$	q_1
q_1	ϕ	$\{q_0,q_1\}$

We will obtain δ' transition for state q_0

$\delta'(q_0,0) = \delta(\{q_0,q_1\},0)$

$= \delta(q_0,0) \cup \delta(q_1,0)$

$= \{q_0,q_1\} \cup \phi$ (new state)

$\delta'(q_0,1) = \delta(q_1,1)$

$= (q_0,q_1)$ (new state.

	0	1
q_0	$\{q_0,q_1\}$	q_1
q_1	ϕ	$\{q_0,q_1\}$
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_1\}$
ϕ	ϕ	ϕ

We will obtain δ' transition for state q_1

$\delta'[q\ \delta'(q_1,0)] = \phi$

$\delta'(q_1,1) = \delta\{(q_0,q_1),1\}$

$= \delta(q_0,1) \cup \delta(q_1,1)$

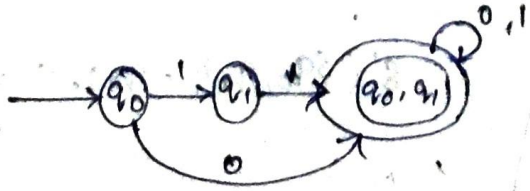
$= q_1 \cup \{q_0,q_1\}$

$\delta((q_0,q_1),0) = (q_0,0) \cup (q_1,0)$

$= (q_0,q_1) \cup \phi$

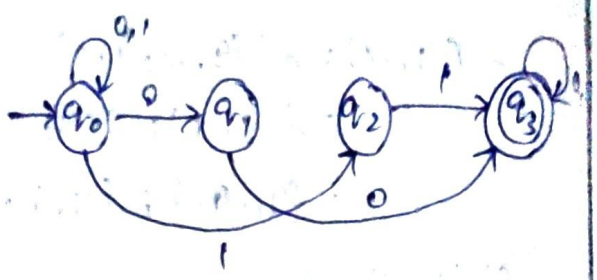
$\delta((q_0,q_1),1) = \delta(q_0,1) \cup \delta(q_1,1)$

$= \{q_1\} \cup \{q_0,q_1\}$



*

	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0, q_2\}$
q_1	q_3	ϕ
q_2	ϕ	q_3
(q_3)	q_3	q_3



We will obtain δ' transition for state q_0

$$\delta'(q_0, 0) = \delta(\{q_0, q_1\}, 0)$$
$$= \delta(q_0, 0) \cup \delta(q_1, 0)$$
$$= \{q_0, q_1\} \cup \{q_3\}$$
$$\delta'(q_0, 1) = \delta(\{q_0, q_2\}, 1)$$
$$= \delta(q_0, 1) \cup \delta(q_2, 1)$$
$$= \{q_0, q_2\} \cup q_3$$

We will obtain δ' transition for state $\{q_0, q_1\}$

$$\delta'(\{q_0, q_1\}, 0) = \delta\{q_0, 0\} \cup \delta\{q_1, 0\}$$
$$= \{q_0, q_1\} \cup q_3$$
$$\delta'(\{q_0, q_1\}, 1) = \delta\{q_0, 1\} \cup \delta\{q_1, 1\}$$
$$= \{q_0, q_2\} \cup \phi$$

We will obtain δ' transition for state $\{q_0, q_2\}$

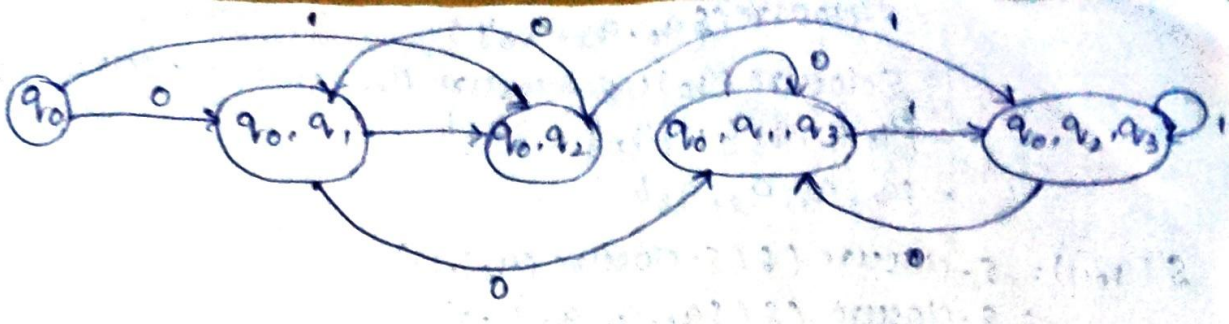
$$\delta'(\{q_0, q_2\}, 0) = \delta\{(q_0, 0) \cup \delta(q_2, 0)\}$$
$$= \{q_0, q_1\} \cup \phi$$
$$\delta'(\{q_0, q_2\}, 1) = \delta(q_0, 1) \cup \delta(q_2, 1)$$
$$= \{q_0, q_2\} \cup q_3$$

We will obtain δ' transition q_0, q_1, q_3.

$$\delta'((\{q_0, q_1\}, q_3), 0) = \delta\{q_0, q_1\} \cup \delta(q_3, 0)$$
$$= \{q_0, q_1\} \cup q_3$$
$$= \{q_0, q_1\} \cup q_3$$
$$\delta'(\{q_0, q_1\}, q_3), 1) = \delta(q_0, q_1) \cup \delta(q_3, 1)$$
$$= \{q_0, q_2\} \cup q_3$$

We will obtain δ' transition for state $\{q_0, q_2\} q_3$

$$\delta'((\{q_0, q_1\}, q_3), 0) = \delta(\{q_0, q_1\}, 0) \cup \delta(q_3, 0)$$
$$= \{q_0, q_1\} \cup q_3$$
$$\delta'(\{q_0, q_1\}, q_3), 1) = \delta(\{q_0, q_1\}, q_3), 1)$$
$$= \delta((q_0, q_1), 1) \cup \delta(q_3, 1)$$
$$= \{q_0, q_2\} \cup q_3$$



state	inputs	
	0	1
q_0	q_0, q_1	q_0, q_2
q_0, q_1	q_0, q_1, q_3	q_0, q_2
q_0, q_2	q_0, q_1	q_0, q_2, q_3
q_0, q_1, q_3	q_0, q_1, q_3	q_0, q_2, q_3
q_0, q_2, q_3	q_0, q_1, q_3	q_0, q_2, q_3

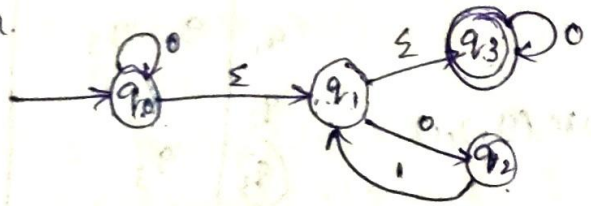
* conversion NFA with Σ to NFA:

let $M = \{Q, \Sigma, \delta, q_0, F\}$ is NFA with Σ then NFA is denoted by $M' = \{Q_0, \Sigma, \delta', q_0, F\}$

procedure:

1. We will take Σ-closure for all states.
2. For each state find transitions by Σ-closure of state, given input and then finally Σ-closure the result. this eliminates Σ-closures.
3. Final state will be the all states contain F.

ex: convert NFA with Σ-moves to NFA for the given finite automata.



state	input		
	0	1	Σ
q_0	q_0	ϕ	q_1
q_1	q_2	ϕ	q_3
q_2	ϕ	q_2	ϕ
q_3	q_3	ϕ	ϕ

Σ-closure of $(q_0) = \{q_0, q_1, q_3\}$

Σ-closure of $(q_1) = \{q_1, q_3\}$

Σ-closure of $(q_2) = \{q_2\}$

Σ-closure of $(q_3) = \{q_3\}$

$\delta'(q_0, 0) = \Sigma$-closure $(\delta(\Sigma$-closure $(q_0), 0))$

$= \Sigma$-closure $(\delta(\{q_0, q_1, q_3\}, 0))$

$= \Sigma$-closure $(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_3, 0))$

$= \Sigma$-closure $(\{q_0\} \cup \{q_2\} \cup \{q_3\})$

$= \Sigma\text{-closure}(\{q_0, q_2, q_3\})$

$= \Sigma\text{closure}(q_0) \cup \Sigma\text{-closure}(q_2) \cup \Sigma\text{-closure}(q_3)$

$= \{q_0, q_1, q_3\} \cup \{q_2\} \cup \{q_3\}$

$= \{q_0, q_1, q_2, q_3\}$

$\delta'(q_0, 1) = \Sigma\text{-closure}(\delta(\Sigma\text{-closure}(q_0), 1))$

$= \Sigma\text{-closure}(\delta(\{q_0, q_1, q_3\}, 1)$

$= \Sigma\text{-closure}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_3, 1))$

$= \Sigma\text{-closure}(\emptyset \cup \emptyset \cup \emptyset)$

$= \Sigma\text{-closure}(\emptyset)$

$= \emptyset$

$\delta'(q_1, 0) = \Sigma\text{-closure}(\delta(\Sigma\text{-closure}(q_1), 0)$

$= \Sigma\text{-closure}(\delta\{q_1, q_3\}, 0)$

$= \Sigma\text{-closure}(\delta(q_1, 0) \cup \delta(q_3, 0))$

$= \Sigma\text{-closure}(\{q_2\} \cup \{q_3\})$

$= \Sigma\text{-closure}(\{q_2, q_3\})$

$= \Sigma\text{-closure}(q_2) \cup \Sigma\text{-closure}(q_3)$

$= \{q_2\} \cup \{q_3\} = \{q_2, q_3\}$

$\delta'(q_1, 1) = \Sigma\text{-closure}(\delta(\Sigma\text{-closure}(q_1), 1)$

$= \Sigma\text{-closure}(\delta\{q_1, q_3\}, 1)$

$= \Sigma\text{-closure}(\delta(q_1, 1) \cup \delta(q_3, 1))$

$= \Sigma\text{-closure}(\emptyset \cup \emptyset)$

$= \Sigma\text{-closure}(\emptyset)$

$= \emptyset$

$\delta'(q_2, 0) = \Sigma\text{-closure}(\delta(\Sigma\text{-closure}(q_2), 0)$

$= \Sigma\text{-closure}(\delta\{q_2\}, 0)$

$= \Sigma\text{-closure}(\delta(q_2, 0))$

$= \Sigma\text{-closure}(\emptyset)$

$= \emptyset$

$\delta'(q_2, 1) = \Sigma\text{-closure}(\delta(\Sigma\text{-closure}(q_2), 1)$

$= \Sigma\text{-closure}(\delta\{q_1\}, 1)$

$= \Sigma\text{-closure}\{q_1, q_3\}$

$= \{q_1, q_3\}$

$\delta'(q_3, 0) = \Sigma\text{-closure}(\delta(\Sigma\text{-closure}(q_3), 0)$

$= \Sigma\text{-closure}(\delta\{q_3\}, 0)$

$= \Sigma\text{-closure}(\delta(q_3, 0))$

$= q_3$

$\delta'(q_3, 1) = \Sigma\text{-closure}(\delta(\Sigma\text{-closure}(q_3), 1)$

$= \Sigma\text{-closure}(\delta\{q_3\}, 1)$

$= \Sigma\text{-closure}(\delta(q_3, 1))$

$= \emptyset$

	0	1
$\rightarrow q_0$	q_0, q_1, q_2, q_3	$\emptyset$
q_1	q_2, q_3	$\emptyset$
q_2	$\emptyset$	q_1, q_3
(q_3)	q_3	$\emptyset$

	0	1	2
q_0	q_0, q_1, q_2	q_1, q_2	q_2
q_1	ϕ	q_1, q_2	q_2
q_2	ϕ	ϕ	q_2

ε-closure of $q_0 = \{q_0, q_1, q_2\}$
ε-closure of $q_1 = \{q_1, q_2\}$
ε-closure of $q_2 = \{q_2\}$

$\delta'(q_0,0)$ = ε-closure $\delta(q_0,0)$
= ε-closure $\delta\{(q_0,q_1,q_2),0\}$
= ε-closure $\delta(q_0,0) \cup \delta(q_1,0) \cup \delta(q_2,0)$
= ε-closure $\{q_0 \cup \phi \cup \phi\}$
= ε-closure $\{q_0\} = q_0 q_1 q_2$

$\delta'(q_0,1)$ = ε-closure $(\delta(q_0,1))$
= ε-closure $(\delta\{\{q_0,q_1,q_2\},1\})$
= ε-closure $\{\phi \cup q_1 \cup \phi\}$
= ε-closure $\{q_1\}$
= $q_1 q_2$

$\delta'(q_0,2)$ = ε-closure $\delta(q_0,2)$
= ε-closure $(\delta\{q_0 q_1 q_2\},2)$
= q_2

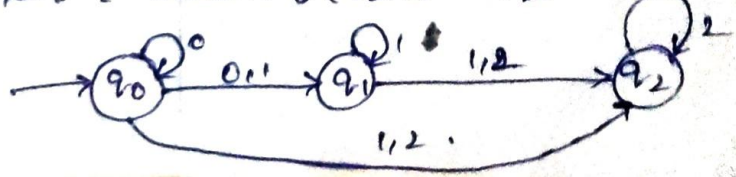
$\delta'(q_1,0)$ = ε-closure $\delta(q_1,0)$
= ε-closure $(\delta((q_1,q_2),0))$
= ε-closure $(\delta\{(q_1,0) \cup (q_2,0)\})$
= ε-closure $(\phi \cup \phi)$
= ϕ

$\delta'(q_1,1)$ = ε-closure $\delta(q_1,1)$
= ε-closure $(\delta((q_1,q_2),1))$
= ε-closure $(\delta\{(q_1,1) \cup (q_2,1)\})$
= ε-closure $(q_1 \cup \phi)$
= ε-closure (q_1)
= q_1, q_2.

$\delta'(q_1,2)$ = ε-closure $(\delta(q_1,2))$
= ε-closure $(\delta(\{q_1,q_2\},2))$
= ε-closure $(\delta\{(q_1,2) \cup (q_2,2)\})$
= ε-closure $(\phi \cup q_2)$
= ε-closure (q_2)
= q_2

State	0	1	2	ε
q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
q_2	ϕ	ϕ	q_2	ϕ

$\delta'(q_2,0)$ = ε-closure $\delta(q_2,0)$ = ε-closure $(\delta(q_2,0)) = \phi$
$\delta'(q_2,1)$ = ε-closure $\delta(q_2,1) = \phi$
$\delta'(q_2,2)$ = ε-closure $\delta(q_2,2) = q_2$



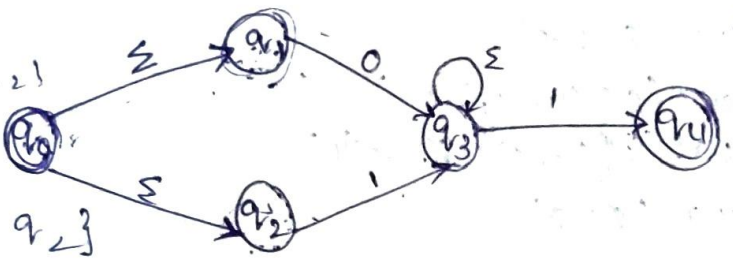
* Conversion of NFA with Σ to DFA:

let $M = \{Q, \Sigma, \delta, q_0, F\}$ is NFA with Σ then DFA is denoted by $M' = \{Q', \Sigma', q_0, \delta', F'\}$

procedure:

1. We will take Σ-closure for the starting state of NFA as start state of DFA.
2. Find states for each input sign symbol that can be traversed from present i.e. union of transition value and their closures for each state of NFA present in the current state of DFA.
3. If we found new state, take it as current state and repeat step 2.
4. Repeat step 2 and step 3 until there is no new state present in transition table of DFA.
5. Mark states of DFA as final state which contains final state of NFA

Ex:



We will obtain δ' transition for starting state $\{q_0, q_1, q_2\}$

$\delta'(\{q_0, q_1, q_2\}, 0) = \Sigma\text{-closure}(\delta(\{q_0, q_1, q_2\}, 0))$

$= \Sigma\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$

$= \Sigma\text{-closure}(\emptyset \cup q_3 \cup \emptyset)$

$= \Sigma\text{-closure}(q_3)$

$= \{q_3\}$

$\delta'(\{q_0, q_1, q_2\}, 1)$

$= \Sigma\text{-closure}(\delta(\{q_0, q_1, q_2\}, 1))$

$= \Sigma\text{-closure}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1))$

$= \Sigma\text{-closure}(\emptyset \cup \emptyset \cup q_3)$

$= \Sigma\text{-closure}(q_3)$

$= \{q_3\}$

	0	1
$\rightarrow q_0, q_1, q_2$	q_3	q_3
q_3	$\emptyset$	q_4
q_4	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$

$\delta'(\{q_3\},0)$

We will obtain δ' transition for ~~starting~~ state $\{q_3\}$

$\delta'(\{q_3\},0) = \varepsilon\text{-closure}(\delta(\{q_3\},0)$

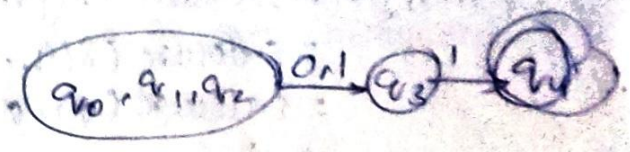
$= \varepsilon\text{-closure}(\delta(q_3,0))$

$= \{\emptyset\}$.

$\delta'(q_3,1) = \varepsilon\text{-closure}(\delta\{q_3\},1)$

$= \varepsilon\text{-closure}(\delta(q_3,1))$

$= q_4$



$\delta'(q_4,0) = \varepsilon\text{-closure}(\delta\{q_4\},0)$

$= \varepsilon\text{-closure}(\delta(q_4,0))$

$= \{\emptyset\}$

$\delta'(q_4,1) = \varepsilon\text{-closure}(\delta(\{q_4\},1))$

$= \varepsilon\text{-closure}(\delta(q_4,1))$

$= \{\emptyset\}$

$\delta'(\emptyset,0) = \varepsilon\text{-closure}(\delta\{\emptyset\},0)$

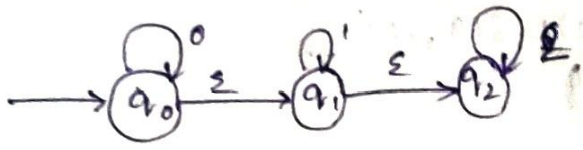
$= \{\emptyset\}$

$\delta'(\emptyset,1) = \varepsilon\text{-closure}(\delta\{\emptyset\},1)$

$= \varepsilon\text{-closure}(\emptyset,1)$

$= \emptyset$

*



ε-closure of $(q_0) = \{q_0, q_1, q_2\}$

ε-closure of $(q_1) = \{q_1, q_2\}$

ε-closure of $(q_2) = \{q_2\}$

We will obtain δ' transition for starting state $\{q_0, q_1, q_2\}$

	0	1	2
q_0, q_1, q_2	q_0, q_1, q_2	q_1, q_2	q_2
q_1, q_2	$\emptyset$	q_1, q_2	q_2
q_2	$\emptyset$	$\emptyset$	q_2
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

$\delta'(\{q_0, q_1, q_2\}, 0) = \varepsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 0)$

$= \varepsilon\text{-closure}(\delta(q_0,0) \cup \delta(q_1,0) \cup \delta(q_2,0))$

$= \varepsilon\text{-closure}(q_0 \cup \emptyset \cup \emptyset)$

$= \varepsilon\text{-closure}\{q_0\}$

$= \{q_0, q_1, q_2\}$

$\delta'(\{q_0, q_1, q_2\}, 1) = \varepsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}), 1)$

$= \varepsilon\text{-closure}(\delta(q_0,1) \cup \delta(q_1,1) \cup \delta(q_2,1))$

$= \varepsilon\text{-closure}(\emptyset \cup q_1 \cup \emptyset)$

$= \varepsilon\text{-closure}(q_1)$

$= \{q_1, q_2\}$

$\delta'(\{q_0, q_1, q_2\}, 2) = \Sigma\text{-closure}(\delta(\{q_0, q_1, q_2\}, 2))$

$= \Sigma\text{-closure}(\delta(q_0, 2) \cup \delta(q_1, 2) \cup \delta(q_2, 2))$

$= \Sigma\text{-closure}(\phi \cup \phi \cup q_2)$

$= \Sigma\text{-closure}(q_2)$

$= q_2$

$\delta'(\{q_1, q_2\}, 0) = \Sigma\text{-closure}(\delta(\{q_1, q_2\}), 0)$

$= \Sigma\text{-closure}(\delta(q_1, 0) \cup \delta(q_2, 0))$

$= \Sigma\text{-closure}(\phi \cup \phi)$

$= \{\phi\}$

$\delta'(\{q_1, q_2\}, 1) = \Sigma\text{-closure}(\delta(\{q_1, q_2\}), 1)$

$= \Sigma\text{-closure}(\delta(q_1, 1) \cup \delta(q_2, 1))$

$= \Sigma\text{-closure}(q_1 \cup \phi)$

$= \Sigma\text{-closure}(q_1)$

$= \{q_1, q_2\}$

$\delta'(\{q_1, q_2\}, 2) = \Sigma\text{-closure}(\delta(\{q_1, q_2\}), 2)$

$= \Sigma\text{-closure}(\delta(q_1, 2) \cup \delta(q_2, 2))$

$= \Sigma\text{-closure}(\phi \cup q_2)$

$= \Sigma\text{-closure}(q_2)$

$= q_2$

$\delta'(q_2, 0) = \Sigma\text{-closure}(\delta(\{q_2\}), 0)$

$= \Sigma\text{-closure}(\delta(q_2, 0))$

$= \Sigma\text{-closure}(\phi)$

$= \{\phi\}$

$\delta'(q_2, 1) = \Sigma\text{-closure}(\delta(q_2, 1))$

$= \Sigma\text{-closure}(\phi)$

$= \{\phi\}$

$\delta'(q_2, 2) = \Sigma\text{-closure}(\delta(q_2, 2))$

$= \Sigma\text{-closure}(q_2)$

$= \{q_2\}$

$\delta'(\phi, 0) = \Sigma\text{-closure}(\delta(\phi, 0))$

$= \{\phi\}$

$\delta'(\phi, 1) = \Sigma\text{-closure}(\delta(\phi, 1))$

$= \{\phi\}$

$\delta'(\phi, 2) = \Sigma\text{-closure}(\delta(\phi, 2))$

$= \{\phi\}$

* Minimization of finite Automata:

Minimization of Automata means reducing the no. of States from given Finite Automata. Minimization of finite Automata can be done in 2 ways.

1. Equivalance / partition method
2. Table Filling method / myhill Nerode theorem.

1. Equivalance / partition method:

let $M = \{Q, \Sigma, \delta, q_0, F\}$ is DFA or NFA then minimized DFA or NFA is denoted by $M' = \{Q', \Sigma, \delta', q_0, F'\}$

M is for given input
M' is for output.

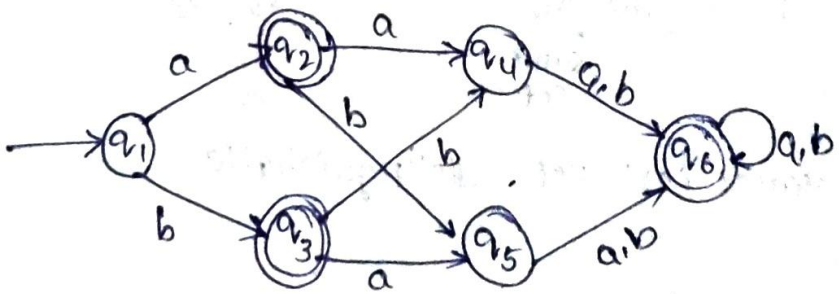
Procedure:-

1. We will divide total states into 2 sets. one set contains all final states and other set contains all non final states. This is called zero-equivalance (or) partition P_0.
2. Find one-equivalance (or) partition P_1 from previous partition P_0 by checking all possible pairs of states in a set of distinguishable R_0. If two states are distinguishable we split the set into two different sets.

Note: If two sets are distinguishable means for any input symbol a the transaction results are in different set of partitions.

3. Repeat step 2 until no change in 2 consecutive equivalance (or) partitions.
4. All states of one set are merged into one state.

* Minimize the given DFA by equivalance (or) partition method.



States = $\{q_1, q_2, q_3, q_4, q_5, q_6\}$

0-equivalance = $\{q_1, q_4, q_5\}$ $\{q_2, q_3, q_6\}$
Non Final Final.

(q_1, q_4)

$(q_1, a) = q_2$ | $(q_1, b) = q_3$

$(q_2, a) = q_6$ | $(q_2, b) = q_2$

same set | same set

$q_1 = q_4$

q_1 & q_4 are equivalance (or) not distinguishable.

(q_4, q_5)

$(q_4, a) = q_6$ | $(q_4, b) = q_6$

$(q_5, a) = q_6$ | $(q_5, b) = q_6$

same set | same set

$q_4 = q_5$

q_4 & q_5 are equivalance (or) not distinguishable.

(q_1, q_5)

$(q_1, a) = q_2$ | $(q_5, b) = q_3$

$(q_5, a) = q_6$ | $(q_5, b) = q_6$

same set | same set

$q_1 = q_5$

q_1, q_5 are equivalance (or) not distinguishable.

$\{q_2, q_3, q_6\}$

(q_2, q_3)

$(q_2, a) = q_4$ | $(q_2, b) = q_5$

$(q_3, a) = q_5$ | $(q_3, b) = q_4$

same set | same set

$q_2 = q_3$

q_2 & q_3 are equivalance (or) not distinguishable.

(q_3, q_6)

$(q_3, a) = q_5$ | $(q_3, b) = q_4$

$(q_6, a) = q_6$ | $(q_6, b) = q_6$

$q_3 \neq q_6$

$q_2 \neq q_6$

Not equivalance.

$\{q_2, q_3\}$ $\{q_6\}$

(q_2, q_6)

$(q_2, a) = q_4$	$(q_2, b) = q_5$
$(q_6, a) = q_6$	$(q_6, b) = q_6$

Not equivalance.

Finally one equivalance is

$\{q_1, q_4, q_5\}$ $\{q_2, q_3\}$ $\{q_6\}$

q_1, q_4

$(q_1, a) = q_2$	$(q_1, b) = q_3$
$(q_4, a) = q_6$	$(q_4, b) = q_2$

$q_1 \neq q_4$

q_1, q_4 are not equivalant

q_4, q_5

$(q_4, a) = q_6$	$(q_5, a) = q_6$
$(q_5, a) = q_6$	$(q_5, a) = q_6$

$q_4 = q_5$

q_4, q_5 are equivalant

q_1, q_5

$(q_1, a) = q_2$	$(q_5, b) = q_3$
$(q_5, a) = q_6$	$(q_5, b) = q_6$

$q_1 \neq q_5$

$\{q_4, q_5\}$ $\{q_1\}$

q_1 & q_5 are not equivalant

(q_2, q_3)

$(q_2, a) = q_4$	$(q_2, b) = q_5$
$(q_3, a) = q_5$	$(q_3, b) = q_4$

$q_2 = q_3$ $\{q_6\}$

q_2, q_3 are ~~not~~ equivalent.

(q_6)

$(q_6, a) = q_6$

$(q_6, b) = q_6$

q_6 is equivalent

Finally two equivalant

$\{q_1\}$ $\{q_4, q_5\}$ $\{q_2, q_3\}$ $\{q_6\}$

(q_4, q_5)

$(q_4, a) = q_6$	$(q_4, b) = q_6$
$(q_5, a) = q_6$	$(q_5, b) = q_6$.

$q_4 = q_5$

q_4 & q_5 are equivalant.

(q_2, q_3)

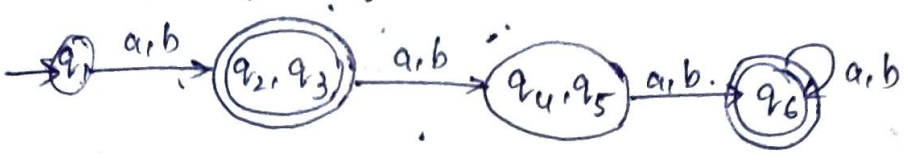
$(q_2, a) = q_4$	$(q_2, b) = q_5$
$(q_3, a) = q_5$	$(q_3, b) = q_4$

$q_2 = q_3$

q_4 & q_5 are equivalant.

Finally three equivalant

$\{q_1\}$ $\{q_4, q_5\}$ $\{q_2, q_3\}$ $\{q_6\}$



*Equivalance:-

The two states q_1, q_2 are equivalant if $\delta(q_1, x)$ and $\delta(q_2, x)$ are either final states or non final states.

2. Table Filling method (or) My hill nerode theorem:

let $M = \{Q, \Sigma, \delta, q_0, F\}$ is DFA or NFA then minimized DFA or NFA is denoted by $M' = \{Q', \Sigma, \delta', q_0, F'\}$

procedure:-

① Draw a table for all pair of states (P,Q)

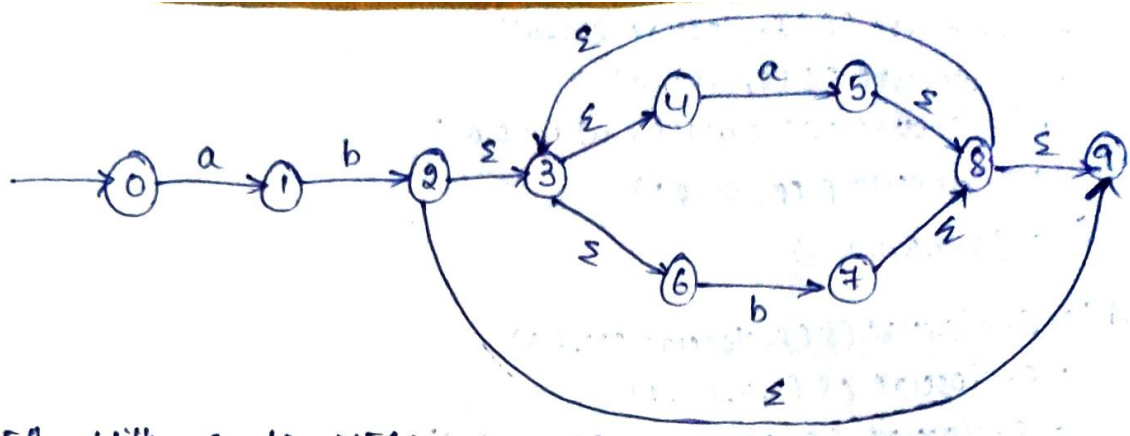
② Mark all pairs where $P \in F$ and $Q \notin F$

Where F = Final state.

③ If there are any unmarked pairs (P,Q) such that $[\delta(P, x), \delta(Q, x)]$ then mark (P,Q) with x as input symbol.

Repeat step 3 until no more marking can be done.

④ combine all unmarked pairs making them as a single state



NFA With ε to NFA:

ε-closure of 0 = {0}
~~ε-closure of q_2 = {2, q_3, q_4, q_6, q_9}~~
ε-closure of 1 = {1}
ε-closure of 2 = {2,3,4,6,9}
ε-closure of 3 = {3,4,6}
ε-closure of 4 = {4}
ε-closure of 5 = {3,4,5,6,8,9}
ε-closure of 6 = {6}
ε-closure of 7 = {3,4,6,7,8,9}
ε-closure of 8 = {3,4,6,8,9}
ε-closure of 9 = {9}

$\delta'(0,a)$ = ε-closure(δ(ε-closure(0), a))
= ε-closure(δ(0,a))
= ε-closure(1)
= {1}

$\delta'(0,b)$ = ε-closure(δ(ε-closure(0), b))
= ε-closure(δ(0,b))
= ε-closure(∅)
= {∅}

$\delta'(1,a)$ = ε-closure(δ(ε-closure(1), a))
= ε-closure(δ(1,a))
= ε-closure(∅)
= {∅}

$\delta'(1,b)$ = ε-closure(δ(ε-closure(1), b))
= ε-closure(δ(1,b))
= ε-closure(2)
= {2,3,4,6,9}

$\delta'(2,a)$ = ε-closure(δ(ε-closure(2), a))
= ε-closure(δ(2,3,4,6,9), a)
= ε-closure(δ(2,a) ∪ (3,a) ∪ (4,a) ∪ (6,a) ∪ (9,a))
= ε-closure(∅ ∪ ∅ ∪ 5 ∪ ∅ ∪ ∅)
= {3,4,5,6,8,9}

$\delta'(2,b)$ = ε-closure(δ(ε-closure(2), b))
= ε-closure(δ {2,3,4,6,9}, b)
= ε-closure(δ(2,b) ∪ (3,b) ∪ (4,b) ∪ (6,b) ∪ (9,b))
= ε-closure(∅ ∪ ∅ ∪ ∅ ∪ 7 ∪ ∅)
= {3,4,6,7,8,9}

	a	b
1	1	∅
2	∅	2,3,4,6,9
3	3,4,5,6,8,9	3,4,6,7,8
4	3,4,5,6,8,9	∅
5	3,4,5,6,8,9	3,4,6,7,8,
6	∅	3,4,6,7,8,9
7	3,4,5,6,8,9	3,4,6,7,8,
8	3,4,5,6,8,9	3,4,6,8,9
9	∅	∅

$\delta'(3,a) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(3),a))$

$= \varepsilon\text{-closure}(\delta(3,4,6),a)$

$= \varepsilon\text{-closure}(\delta((3,a)\cup(4,a)\cup(6,a))$

$= \varepsilon\text{-closure}((\phi\cup 5\cup\phi))$

$= \{3,4,5,6,8,9\}$

$\delta'(3,b) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(3),b))$

$= \varepsilon\text{-closure}(\delta(3,4,6),b)$

$= \varepsilon\text{-closure}(\delta((3,b)\cup(4,b)\cup(6,b)))$

$= \varepsilon\text{-closure}(\phi\cup\phi\cup 7)$

$= \{3,4,6,7,8,9\}$

$\delta'(4,a) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(4),a))$

$= \varepsilon\text{-closure}(\delta(4,a))$

$= \varepsilon\text{-closure}(5)$

$= \{3,4,5,6,8,9\}$

$\delta'(4,b) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(4),b))$

$= \varepsilon\text{-closure}(\delta(4,b))$

$= \varepsilon\text{-closure}(\phi)$

$= \{\phi\}$

$\delta'(5,a) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(5),a))$

$= \varepsilon\text{-closure}(\delta\{3,4,5,6,8,9\},a)$

$= \varepsilon\text{-closure}(\delta((3,a)\cup(4,a)\cup(5,a)\cup(6,a)\cup(8,a)\cup(9,a)))$

$= \varepsilon\text{-closure}(\phi\cup 5\cup\phi\cup\phi\cup\phi)$

$= \{3,4,5,6,8,9\}$

$\delta'(5,b) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(5),b))$

$= \varepsilon\text{-closure}(\delta(\{3,4,5,6,7,8,9\},b))$

$= \varepsilon\text{-closure}(\delta((3,b)\cup(4,b)\cup(5,b)\cup(6,b)\cup(7,b)\cup(8,b)\cup(9,b)))$

$= \varepsilon\text{-closure}((\phi\cup\phi\cup\phi\cup 7\cup\phi\cup\phi))$

$= \{3,4,6,7,8,9\}$

$\delta'(6,a) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(6),a))$

$= \varepsilon\text{-closure}(\delta(6,a))$

$= \varepsilon\text{-closure}(\phi)$

$= \{\phi\}$

$\delta'(6,b) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(6),b))$

$= \varepsilon\text{-closure}(\delta(6,b))$

$= \varepsilon\text{-closure}(7)$

$= \{3,4,6,7,8,9\}$

$\delta'(7,a) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(7),a))$

$= \varepsilon\text{-closure}(\delta(\{3,4,6,7,8,9\},a))$

$= \varepsilon\text{-closure}(\delta((3,a)\cup(4,a)\cup(6,a)\cup(7,a)\cup(8,a),(9,a)))$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup \phi \cup \phi \cup \phi)$

$= \varepsilon\text{-closure}(\phi)$

$= \{3,4,5,6,8,9\}$

$\delta'(7,b) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(7),b))$

$= \varepsilon\text{-closure}(\delta(\{3,4,6,7,8,9\},b)$

$= \varepsilon\text{-closure}(\delta((3,b) \cup (4,b) \cup (6,b) \cup (7,b) \cup (8,b) \cup (9,b)))$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup 7 \cup \phi \cup \phi \cup \phi)$

$= \{3,4,6,7,8,9\}$

$\delta'(8,a) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(8),a))$

$= \varepsilon\text{-closure}(\delta(3,4,6,8,9),a)$

$= \varepsilon\text{-closure}(\delta((3,a) \cup (4,a) \cup (6,a) \cup (8,a) \cup (9,a)))$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup \phi \cup 5 \cup \phi)$

$= \{3,4,5,6,8,9\}$

$\delta'(8,b) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(8),b))$

$= \varepsilon\text{-closure}(\delta(3,8,9),b))$

$= \varepsilon\text{-closure}(\delta((3,b) \cup (8,b) \cup (9,b)))$

$= \varepsilon\text{-closure}(\phi \cup \phi \cup \phi \cup \phi \cup 7)$

$= \{3,4,6,7,8,9\}$

$\delta'(9,a) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(9),a))$

$= \varepsilon\text{-closure}(\delta(9,a))$

$= \varepsilon\text{-closure}(\phi)$

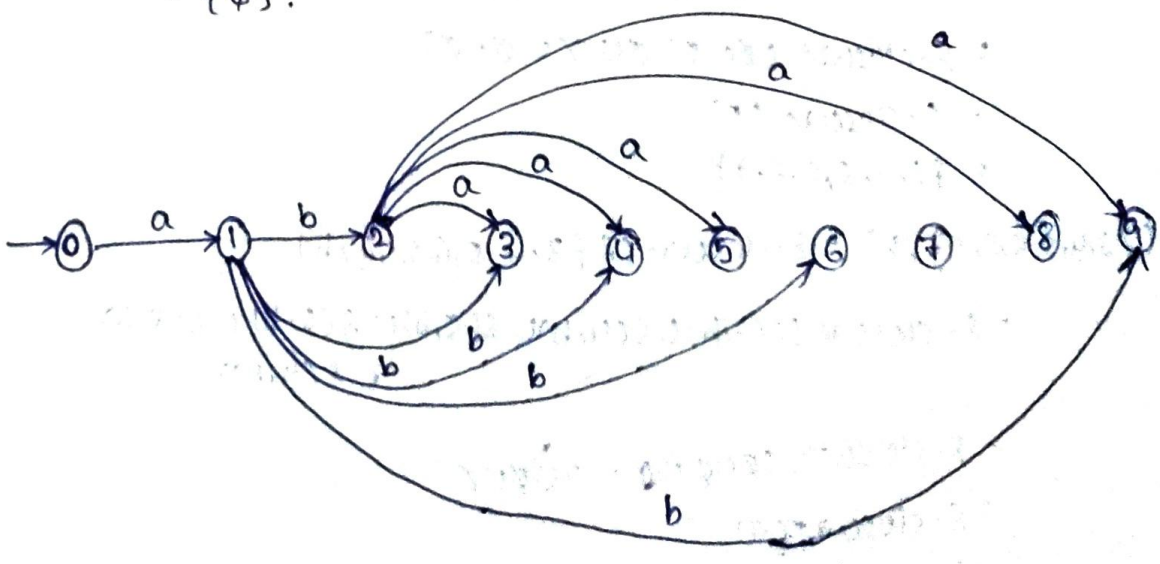
$= \{\phi\}$

$\delta'(9,b) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(9),b))$

$= \varepsilon\text{-closure}(\delta(9,b))$

$= \varepsilon\text{-closure}(\phi)$

$= \{\phi\}$.



NFA with ε to DFA:

ε-closure of 0 = {0}

We will obtain transition for starting state 0.

$\delta'(0,a) = \varepsilon\text{-closure}(\delta(0,a))$
$= \varepsilon\text{-closure}(1)$
$= 1$

$\delta'(0,b) = \varepsilon\text{-closure}(\delta(0,b))$
$= \varepsilon\text{-closure}(\emptyset) = \emptyset$

$\delta'(1,a) = \varepsilon\text{-closure}(\delta(1,a))$
$= \varepsilon\text{-closure}(\emptyset) = \emptyset$

$\delta'(1,b) = \varepsilon\text{-closure}(\delta(1,b))$
$= \varepsilon\text{-closure}(2) = \{2,3,4,6,9\}$

	a	b
→ 0	1	∅
1	∅	{2, 3, 4, 6, 9}
{1, 3, 4, 6, 9}	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}
{3, 4, 5, 6, 8, 9}	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}
{3, 4, 6, 7, 8, 9}	{3, 4, 5, 6, 8, 9}	{3, 4, 6, 7, 8, 9}
∅	∅	∅

$\delta'(\{2,3,4,6,9\},a) = \varepsilon\text{-closure}(\delta\{2,3,4,6,9\},a))$
$= \varepsilon\text{-closure}(\delta(2,a) \cup \delta(3,a) \cup \delta(4,a) \cup \delta(6,a) \cup \delta(9,a))$
$= \varepsilon\text{-closure}(\emptyset \cup \emptyset \cup 5 \cup \emptyset \cup \emptyset)$
$= \varepsilon\text{-closure}(5) = \{3,4,5,6,8,9\}$

$\delta'(\{2,3,4,6,9\},b) = \varepsilon\text{-closure}(\delta\{2,3,4,6,9\},b)$
$= \varepsilon\text{-closure}(\delta(2,b) \cup \delta(3,b) \cup \delta(4,b) \cup \delta(6,b) \cup \delta(9,b))$
$= \varepsilon\text{-closure}(\emptyset \cup \emptyset \cup \emptyset \cup 7 \cup \emptyset)$
$= \varepsilon\text{-closure}(7)$
$= \{3,4,6,7,8,9\}$

$\delta'(\{3,4,5,6,8,9\},a) = \varepsilon\text{-closure}(\delta(\{3,4,5,6,8,9\},a))$
$= \varepsilon\text{-closure}(\delta(3,a) \cup \delta(4,a) \cup \delta(5,a) \cup \delta(6,a) \cup \delta(8,a) \cup \delta(9,a))$
$= \varepsilon\text{-closure}(\emptyset \cup 5 \cup \emptyset \cup \emptyset \cup \emptyset \cup \emptyset)$
$= \varepsilon\text{-closure}(5)$
$= \{3,4,5,6,8,9\}$

$\delta'(\{3,4,5,6,8,9\},b) = \varepsilon\text{-closure}(\delta\{3,4,5,6,8,9\},b)$
$= \varepsilon\text{-closure}(\delta(3,b) \cup \delta(4,b) \cup \delta(5,b) \cup \delta(6,b) \cup \delta(8,b) \cup \delta(9,b))$
$= \varepsilon\text{-closure}(\emptyset \cup \emptyset \cup \emptyset \cup 7 \cup \emptyset \cup \emptyset)$
$= \varepsilon\text{-closure}(7)$
$= \{3,4,6,7,8,9\}$

$\delta'(\{3,4,6,7,8,9\}, a) = \varepsilon\text{-closure}(\delta(\{3,4,6,7,8,9\}, a))$

$= \varepsilon\text{-closure}(\delta(3,a) \cup \delta(4,a) \cup \delta(6,a) \cup \delta(7,a) \cup \delta(8,a) \cup \delta(9,a))$

$= \varepsilon\text{-closure}(\emptyset \cup 5 \cup \emptyset \cup \emptyset \cup \emptyset \cup \emptyset)$

$= \varepsilon\text{-closure}(5)$

$= \{3,4,5,6,8,9\}$

$\delta'(\{3,4,6,7,8,9\}, b) = \varepsilon\text{-closure}(\delta(\{3,4,6,7,8,9\}, b))$

$= \varepsilon\text{-closure}(\delta(3,b) \cup \delta(4,b) \cup \delta(6,b) \cup \delta(7,b) \cup \delta(8,b) \cup \delta(9,b))$

$= \varepsilon\text{-closure}(\emptyset \cup \emptyset \cup 7 \cup \emptyset \cup \emptyset \cup \emptyset)$

$= \varepsilon\text{-closure}(7) = \{3,4,6,7,8,9\}$

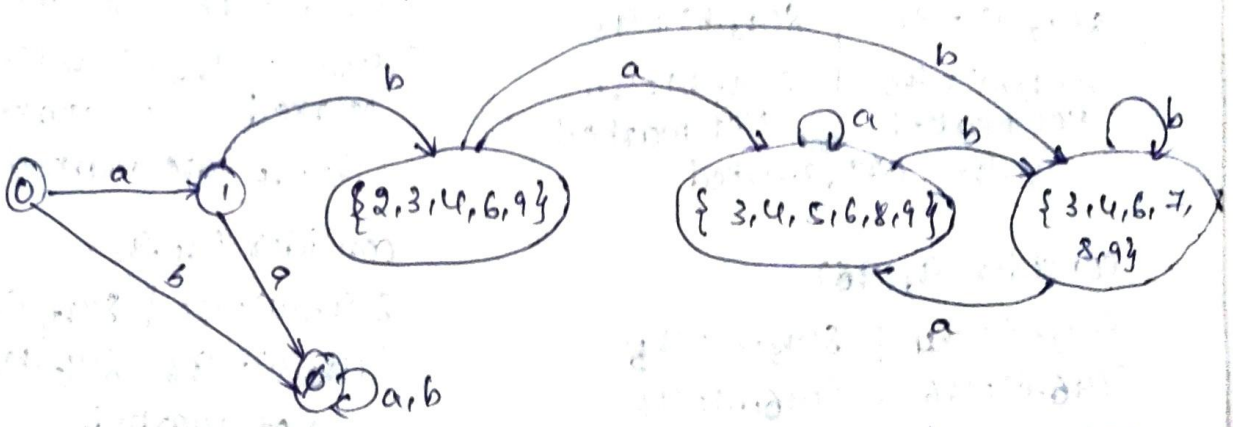
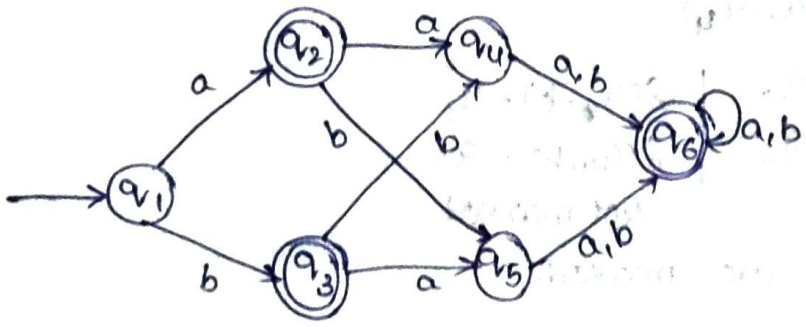
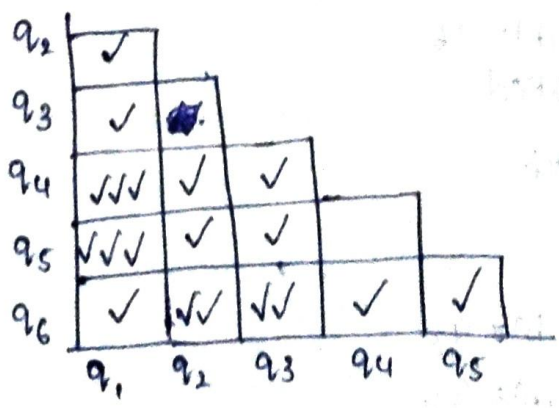


*Table filling method:





consider (q_1, q_4)

$\delta(q_1, a) = q_2$ | $\delta(q_1, b) = q_3$

$\delta(q_4, a) = q_6$ | $\delta(q_4, b) = q_6$

Not marked | Not marked

q_1, q_4 not marked.

consider (q_1, q_5)

$\delta(q_1, a) = q_2$ | $\delta(q_1, b) = q_3$

$\delta(q_5, a) = q_6$ | $\delta(q_5, b) = q_6$

Not marked | Not marked

q_1, q_5 not marked.

consider (q_2, q_3)

$\delta(q_2, a) = q_4$ | $\delta(q_2, b) = q_5$

$\delta(q_3, a) = q_5$ | $\delta(q_3, b) = q_4$

Not marked | Not marked

q_2, q_3 are not marked

consider (q_2, q_6)

$\delta(q_2, a) = q_4$ | $\delta(q_2, b) = q_5$

$\delta(q_6, a) = q_6$ | $\delta(q_6, b) = q_6$

~~not~~ marked | ~~not~~ marked

q_2, q_6 is ~~not~~ marked.

consider (q_1, q_4)

$\delta(q_1, a) = q_2$ | $\delta(q_4, b) = q_3$

$\delta(q_4, a) = q_6$ | $\delta(q_4, b) = q_6$

Marked | Not marked.

q_1, q_4 are marked.

consider q_1, q_5

$\delta(q_1, a) = q_2$ | $\delta(q_1, b) = q_3$

$\delta(q_5, a) = q_6$ | $\delta(q_5, b) = q_6$

marked | marked

q_1, q_5 are marked.

consider q_2, q_3

$\delta(q_2, a) = q_4$ | $\delta(q_2, b) = q_5$

$\delta(q_3, a) = q_5$ | $\delta(q_3, b) = q_4$

Not marked. | Not marked

q_2, q_3 are not marked

consider (q_3, q_6)

$\delta(q_3, a) = q_5$ | $\delta(q_3, b) = q_4$

$\delta(q_6, a) = q_6$ | $\delta(q_6, b) = q_6$

marked | marked

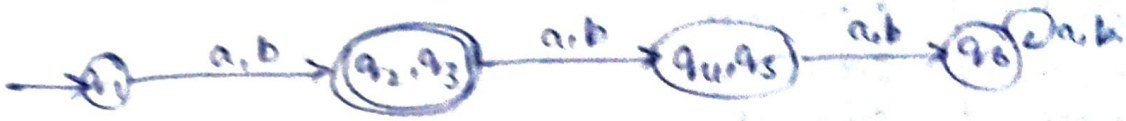
q_3, q_6 are marked

consider (q_4, q_5)

$\delta(q_4, a) = q_6$ | $\delta(q_3, a) = q_6$

$\delta(q_5, a) = q_6$ | $\delta(q_5, b) = q_6$

Not marked.



(1)

	0	1
A	B	E
B	C	D
C	H	I
D	I	H
E	F	G
F	H	I
G	H	I
H	H	H
I	I	I

Equivalence/partition method:

{C, D, F, G} — Final states

{A, B, E, H, I} — non final states.

(C, D)

$\delta(C,0) = H$ | $\delta(C,1) = I$

$\delta(D,0) = I$ | $\delta(D,1) = H$

same set | same set

Equivalent.

(C, F)

$\delta(C,0) = H$ | $\delta(C,1) = I$

$\delta(F,0) = H$ | $\delta(F,1) = I$

same set | same set

C, F are equivalent

(C, G)

$\delta(C,0) = H$ | $\delta(C,1) = I$

$\delta(G,0) = H$ | $\delta(G,1) = I$

same set | same set.

C, G are equivalent

(D, F)

$\delta(D,0) = I$	$\delta(D,1) = H$
$\delta(F,0) = H$	$\delta(F,1) = I$
same set	same set

D, F are equivalent

(D, G)

$\delta(D,0) = I$	$\delta(D,1) = H$
$\delta(G,0) = H$	$\delta(G,1) = I$
same set	same set

D, G are equivalent

(F, G)

$\delta(F,0) = H$	$\delta(F,1) = I$
$\delta(G,0) = H$	$\delta(G,1) = I$
same set	same set

F, G are equivalent.

$\{A, B, E, H, I\}$.

(A, B)

$\delta(A,0) = B$	$\delta(A,1) = E$
$\delta(B,0) = C$	$\delta(B,1) = D$

A, B are not equivalent

(A, E)

$\delta(A,0) = B$	$\delta(A,1) = E$
$\delta(E,0) = F$	$\delta(E,1) = G$.

A, E are not equivalent

(A, H)

$\delta(A,0) = B$	$\delta(A,1) = E$
$\delta(H,0) = H$	$\delta(H,1) = H$
same set	same set

A, H are equivalent

(A, I)

$\delta(A,0) = B$	$\delta(A,1) = E$
$\delta(I,0) = I$	$\delta(I,1) = I$
same set	same set.

A, I are equivalent.

(B, E)

$\delta(B,0) = C$	$\delta(B,1) = D$
$\delta(E,0) = F$	$\delta(E,1) = G$
same set	same set

B, E are equivalent

(B, H)

$\delta(B,0) = C$	$\delta(B,1) = D$
$\delta(H,0) = H$	$\delta(H,1) = H$

B, H are not equivalent

(B, I)

$\delta(B,0) = C$	$\delta(B,1) = D$
$\delta(I,0) = I$	$\delta(I,1) = I$

B, I are not equivalent.

(E, H)

$\delta(E,0) = F$	$\delta(E,1) = G$
$\delta(H,0) = H$	$\delta(H,1) = H$

E, H are not equivalent.

(E, I)

$\delta(E,0) = F$	$\delta(E,1) = G$
$\delta(I,0) = I$	$\delta(I,1) = I$

E, I are not equivalent

(H, I)

$\delta(H,0) = H$	$\delta(H,1) = H$
$\delta(I,0) = I$	$\delta(I,1) = I$
same set	same set

H, I are equivalent

Finally one equivalant is

{A, H, I} {B, E} {C, D, F, G}

(C, D)

$\delta(C,0) = H$	$\delta(C,1) = I$
$\delta(D,0) = I$	$\delta(D,1) = H$
same set	same set.

C, D are equivalent

(C, F)

$\delta(C,0) = H$	$\delta(C,1) = I$
$\delta(F,0) = H$	$\delta(F,1) = I$
same set	same set

C, F are equivalent

(C, G)

$\delta(C,0) = H$	$\delta(C,1) = I$
$\delta(G,0) = H$	$\delta(G,1) = I$
same set	same set

C, G are equivalent

(D, F)

$\delta(D,0) = I$	$\delta(D,1) = H$
$\delta(F,0) = H$	$\delta(F,1) = I$
same set	same set

D, F are equivalent

(D, G)

$\delta(D,0) = I$	$\delta(D,1) = H$
$\delta(G,0) = H$	$\delta(G,1) = I$
same set	same set

D, G are equivalent

(F, G)

$\delta(F,0) = H$	$\delta(F,1) = I$
$\delta(G,0) = H$	$\delta(G,1) = I$
same set	same set

F, G are equivalent

(B, E)

$\delta(B,0) = E$	$\delta(B,1) = D$
$\delta(E,0) = F$	$\delta(E,1) = G$
same set	same set

B, E are equivalent

(A, H)

$\delta(A,0) = B$	$\delta(A,1) = E$
$\delta(H,0) = H$	$\delta(H,1) = H$

A, H are not equivalent

$C \neq G$
$G \neq H$
$C = H$

$C, H \} G$

$\{ G H$

$\{ C D, \{ H,$

$D = C,$
$H \neq D$

(A, I)

$\delta(A,0) = B$ | $\delta(A,1) = E$

$\delta(I,0) = I$ | $\delta(I,1) = I$

A, I are not equivalent

(H, I)

$\delta(H,0) = H$ | $\delta(H,1) = H$

$\delta(I,0) = I$ | $\delta(I,1) = I$

same set | same set

H, I are equivalent.

Finally two equivalent are

{C, D, F, G} {B, E} {A} {H, I}

(C, D)

$\delta(C,0) = H$ | $\delta(C,1) = I$

$\delta(D,0) = I$ | $\delta(D,1) = H$

same set | same set

C, D are equivalent

(C, G)

$\delta(C,0) = H$ | $\delta(C,1) = I$

$\delta(G,0) = H$ | $\delta(G,1) = I$

same set | same set

C, G are equivalent

(D, G)

$\delta(D,0) = I$ | $\delta(D,1) = H$

$\delta(G,0) = H$ | $\delta(G,1) = I$

same set | same set

D, G are equivalent.

(B, E)

$\delta(B,0) = C$ | $\delta(B,1) = D$

$\delta(E,0) = F$ | $\delta(E,1) = G$

same set | same set

B, E are equivalent.

(C, F)

$\delta(C,0) = H$ | $\delta(C,1) = I$

$\delta(F,0) = H$ | $\delta(F,1) = I$

same set | same set.

C, F are equivalent.

(D, F)

$\delta(D,0) = I$ | $\delta(D,1) = H$

$\delta(F,0) = H$ | $\delta(F,1) = I$

same set | same set.

D, F are equivalent

(F, G)

$\delta(F,0) = H$ | $\delta(F,1) = I$

$\delta(G,0) = H$ | $\delta(G,1) = I$

same set | same set

F, G are equivalent.

(H, I)

$\delta(H,0) = H$ | $\delta(H,1) = H$

$\delta(I,0) = I$ | $\delta(I,1) = I$

same set | same set

H, I are equivalent.

Finally three equivalent:

{C, D, F, G} {B, E} {A} {H, I}

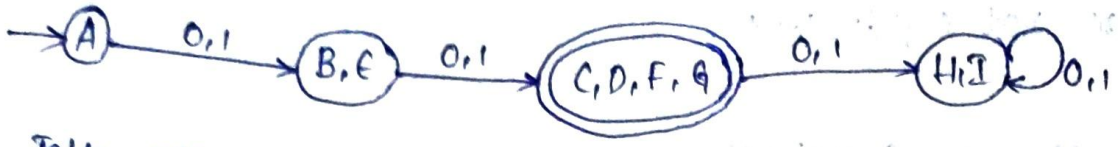
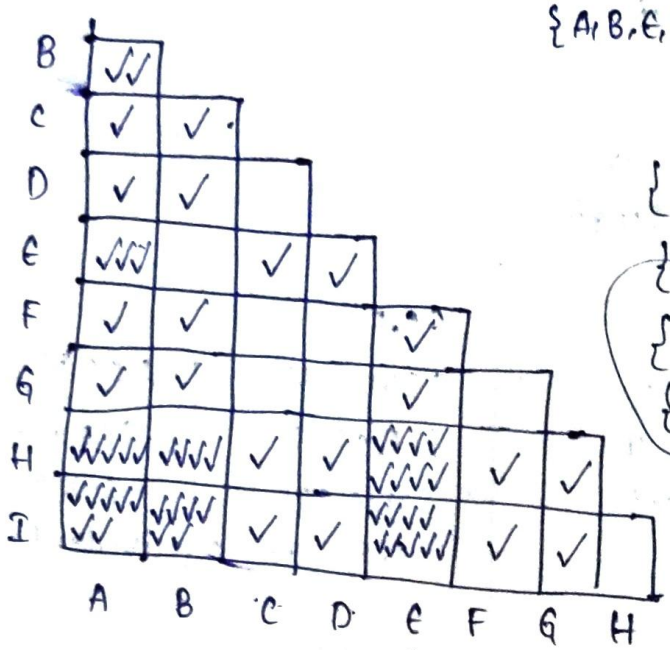


Table Filling method:

{C, D, F, G} = Final

{A, B, E, H, I} = non final.



{B, E} {D, F}

{C, D} {D, G}

{C, F}

{C, G}

consider (A, B)

$\delta(A,0) = B$ | $\delta(A,1) = E$

$\delta(B,0) = C$ | $\delta(B,1) = D$

marked | Marked

A, B are marked.

consider (A, E)

$\delta(A,0) = B$ | $\delta(A,1) = E$

$\delta(G,0) = F$ | $\delta(E,1) = G$

marked | Marked.

A, E are marked.

Consider (A, H)

$\delta(A,0) = B$ | $\delta(A,1) = E$

$\delta(H,0) = H$ | $\delta(H,1) = H$

Not marked | Not marked.

A, I are not marked.

consider (A, I)

$\delta(A,0) = B$ | $\delta(A,1) = E$

$\delta(I,0) = I$ | $\delta(I,1) = I$

Not marked | Not marked

A, I are not marked.

consider (B, E)

$\delta(B,0) = C$ | $\delta(B,1) = D$

$\delta(E,0) = F$ | $\delta(E,1) = G$

Not marked | Not marked

B, E are not marked.

consider (B, H)

$\delta(B,0) = C$ | $\delta(B,1) = D$

$\delta(H,0) = H$ | $\delta(H,1) = H$

marked | marked

B, H are marked.

consider (A, H)

$\delta(A,0) = B$ | $\delta(A,1) = E$

$\delta(H,0) = H$ | $\delta(H,1) = H$

marked | Not marked

A, H are marked.

consider (A, I)

$\delta(A,0) = B$ | $\delta(A,1) = E$

$\delta(I,0) = I$ | $\delta(I,1) = I$

Not marked | not marked

A, I are not marked.

A, I are not marked

consider (B,E)

$\delta(B,0) = C$ | $\delta(B,1) = D$

$\delta(E,0) = F$ | $\delta(E,1) = G$

Not marked | Not marked

B, E are not marked

consider (B,I)

$\delta(B,0) = C$ | $\delta(B,1) = D$

$\delta(I,0) = I$ | $\delta(I,1) = I$

Marked | Marked

B, I are marked.

consider (A,I)

$\delta(A,0) = B$ | $\delta(A,1) = E$

$\delta(I,0) = I$ | $\delta(I,1) = I$

marked | marked

A, I are marked.

consider (B,E)

$\delta(B,0) = C$ | $\delta(B,1) = D$

$\delta(E,0) = F$ | $\delta(E,1) = G$

Not marked | Not marked

B, E are not marked.

consider (C,D)

$\delta(C,0) = H$ | $\delta(C,1) = I$

$\delta(D,0) = I$ | $\delta(D,1) = H$

Not marked | Not marked

C, D are not marked.

consider (C,F)

$\delta(C,0) = H$ | $\delta(C,1) = I$

$\delta(F,0) = H$ | $\delta(F,1) = I$

Not marked | Not marked

C, F are not marked.

consider (C,G)

$\delta(C,0) = H$ | $\delta(C,1) = I$

$\delta(G,0) = H$ | $\delta(G,1) = I$

Not marked | Not marked

C, G are not marked.

consider (D,F)

$\delta(D,0) = I$ | $\delta(D,1) = H$

$\delta(F,0) = H$ | $\delta(F,1) = I$

Not marked | Not marked

D, F are not marked

consider (D,G)

$\delta(D,0) = I$ | $\delta(D,1) = H$

$\delta(G,0) = H$ | $\delta(G,1) = I$

Not marked | Not marked

D, G are not marked.

consider (E,H)

$\delta(E,0) = F$ | $\delta(E,1) = G$

$\delta(H,0) = H$ | $\delta(H,1) = H$.

Marked | marked

E, H are marked.

consider (B,E)

$\delta(B,0) = C$ | $\delta(B,1) = D$

$\delta(E,0) = F$ | $\delta(E,1) = G$

Not marked | Not marked

B, E are not marked.

consider (C,D)

$\delta(C,0) = H$ | $\delta(C,1) = I$

$\delta(D,0) = I$ | $\delta(D,1) = H$

Not marked | Not marked

C, D are not marked

consider (C,F)

$\delta(C,0) = H$ | $\delta(C,1) = I$

$\delta(F,0) = H$ | $\delta(F,1) = I$

Not marked | Not marked.

C, F are not marked.

consider (C,G)

$\delta(C,0) = H$ | $\delta(C,1) = I$

$\delta(G,0) = H$ | $\delta(G,1) = I$

Not marked | Not marked.

C, G are not marked.

consider (E,I)

$\delta(E,0) = F$	$\delta(E,1) = G$
$\delta(I,0) = I$	$\delta(I,1) = I$
Marked	Marked

E, I are marked.

consider (B,E)

$\delta(B,0) = E$	$\delta(B,1) = D$
$\delta(E,0) = F$	$\delta(E,1) = G$
Not marked	Not marked

B, E are not marked

consider (C,D)

$\delta(C,0) = H$	$\delta(C,1) = I$
$\delta(D,0) = I$	$\delta(D,1) = H$
Not marked	Not marked

C, D are not marked.

consider (C,F)

$\delta(C,0) = H$	$\delta(C,1) = I$
$\delta(F,0) = H$	$\delta(F,1) = I$
Not marked	Not marked

C, F are not marked.

consider (C,G)

$\delta(C,0) = H$	$\delta(C,1) = I$
$\delta(G,0) = H$	$\delta(G,1) = I$
Not marked	Not marked

C, G are not marked

consider (D,F)

$\delta(D,0) = I$	$\delta(D,1) = H$
$\delta(F,0) = H$	$\delta(F,1) = I$
Not marked	Not marked

D, F are not marked.

consider (D,G)

$\delta(D,0) = I$	$\delta(D,1) = H$
$\delta(G,0) = H$	$\delta(G,1) = I$
Not marked	Not marked

D, G are not marked.

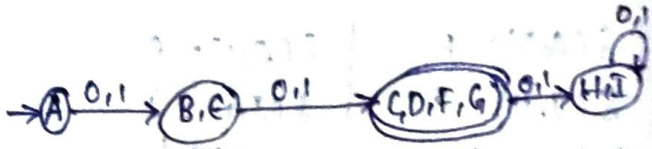
consider (F,G)

$\delta(F,0) = H$	$\delta(F,1) = I$
$\delta(G,0) = H$	$\delta(G,1) = I$
Not marked	Not marked

consider (H,I)

$\delta(H,0) = H$	$\delta(H,1) = H$
$\delta(I,0) = I$	$\delta(I,1) = I$
Not marked	Not marked

H, I are not marked.



(2)

	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_4	q_2
(q_2)	q_1	q_5
q_3	q_0	q_4
q_4	q_4	q_4
q_5	q_2	q_4

Equivalance or partition method:

$\{q_0, q_1, q_3, q_4, q_5\}$ $\{q_2\}$

Non Final States Final states.

(q_0, q_1)

$\delta(q_0,0) = q_1$ | $\delta(q_0,1) = q_3$

$\delta(q_1,0) = q_4$ | $\delta(q_1,1) = q_2$

same set different.

q_0, q_1 are not equivalent

(q_0, q_3)

$\delta(q_0,0) = q_1$ | $\delta(q_0,1) = q_3$

$\delta(q_3,0) = q_0$ | $\delta(q_3,1) = q_4$

same set same set

q_0, q_3 are equivalent.

(q_0, q_4)

$\delta(q_0,0) = q_1$ | $\delta(q_0,1) = q_3$

$\delta(q_3,0) = q_4$ | $\delta(q_4,1) = q_4$

same set same set

q_0, q_4 are equivalent

(q_0, q_5)

$\delta(q_0,0) = q_1$ | $\delta(q_0,1) = q_3$

$\delta(q_5,0) = q_2$ | $\delta(q_5,1) = q_4$

different same set.

q_0, q_5 are not equivalent

(q_1, q_3)

$\delta(q_1,0) = q_4$ | $\delta(q_1,1) = q_2$

$\delta(q_3,0) = q_0$ | $\delta(q_3,1) = q_4$

same set different

q_1, q_3 are not equivalent

(q_1, q_4)

$\delta(q_1,0) = q_4$ | $\delta(q_0,1) = q_2$

$\delta(q_4,0) = q_4$ | $\delta(q_4,1) = q_4$

same set same set

(q_1, q_4) are equivalent

(q_1, q_5)

$\delta(q_1,0) = q_4$ | $\delta(q_1,1) = q_2$

$\delta(q_5,0) = q_2$ | $\delta(q_5,1) = q_4$

different different

(q_1, q_5) are not equivalent

(q_3, q_5)

$\delta(q_3,0) = q_0$ | $\delta(q_3,1) = q_4$

$\delta(q_5,0) = q_2$ | $\delta(q_5,1) = q_4$

different same set

q_3, q_5 are equivalent.

(q_4, q_5)

$\delta(q_4,0) = q_4$ | $\delta(q_4,0) = q_4$ | q_4, q_5 are not equivalent

$\delta(q_5,0) = q_2$ | $\delta(q_5,1) = q_4$

different same

Finally one equivalant.

$\{q_0, q_3\}$ $\{q_1, q_4\}$ $\{q_5\}$ $\{q_2\}$

(q_0, q_3)

$\delta(q_0, 0) = q_1$	$\delta(q_0, 1) = q_3$
$\delta(q_3, 0) = q_0$	$\delta(q_3, 1) = q_4$
different	different

q_0, q_3 are not equivalent

(q_1, q_4)

$\delta(q_1, 0) = q_4$	$\delta(q_1, 1) = q_2$
$\delta(q_4, 0) = q_4$	$\delta(q_4, 1) = q_4$
same	different.

q_1, q_4 are not equivalent.

Finally two equivalent

$\{q_0\}$ $\{q_1\}$ $\{q_2\}$ $\{q_3\}$ $\{q_4\}$ $\{q_5\}$

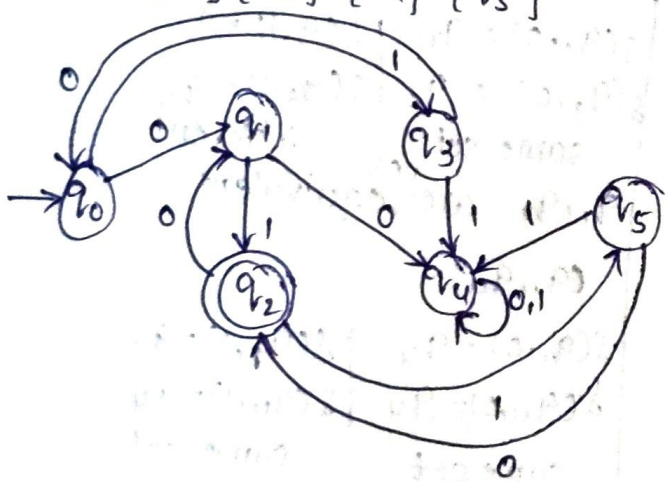
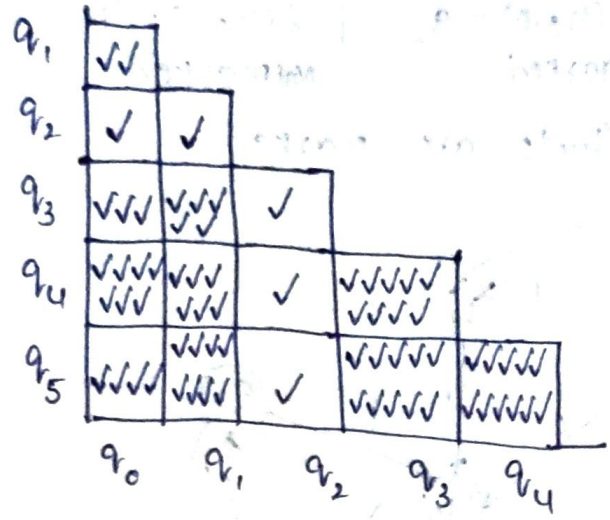


Table Filling method:



$\{q_0, q_1, q_3, q_4, q_5\}$ = Non final

$\{q_2\}$ = Final

consider (q_0, q_1)

$\delta(q_0,0) = q_1$ | $\delta(q_0,1) = q_3$

$\delta(q_1,0) = q_4$ | $\delta(q_1,1) = q_2$

Not marked | marked

q_0, q_1 are marked.

consider (q_0, q_3)

$\delta(q_0,0) = q_1$ | $\delta(q_0,1) = q_3$

$\delta(q_3,0) = q_0$ | $\delta(q_3,1) = q_4$

~~Not~~ marked | Not marked

q_0, q_3 are ~~not~~ marked.

consider (q_0, q_4)

$\delta(q_0,0) = q_1$ | $\delta(q_0,1) = q_3$

$\delta(q_4,0) = q_4$ | $\delta(q_4,1) = q_4$

Not marked | Not marked

q_0, q_4 are not marked

consider (q_0, q_5)

$\delta(q_0,0) = q_1$ | $\delta(q_0,1) = q_3$

$\delta(q_5,0) = q_2$ | $\delta(q_5,1) = q_4$

marked | Not marked

q_0, q_5 are marked.

consider (q_0, q_3)

$\delta(q_0,0) = q_1$ | $\delta(q_0,1) = q_3$

$\delta(q_3,0) = q_4$ | $\delta(q_3,1) = q_4$

Not marked | Not marked

q_0, q_4 are not marked

consider (q_1, q_3)

$\delta(q_1,0) = q_4$ | $\delta(q_1,1) = q_2$

$\delta(q_3,0) = q_0$ | $\delta(q_3,1) = q_4$

Not marked | marked

q_1, q_3 marked.

consider (q_0, q_4)

$\delta(q_0,0) = q_1$ | $\delta(q_0,1) = q_3$

$\delta(q_4,0) = q_4$ | $\delta(q_4,1) = q_4$

Not marked | Not marked

consider (q_1, q_4)

$\delta(q_1,0) = q_4$ | $\delta(q_1,1) = q_2$

$\delta(q_4,0) = q_4$ | $\delta(q_4,1) = q_4$

Not marked | marked

q_1, q_4 are marked.

consider (q_0, q_4)

$\delta(q_0,0) = q_1$ | $\delta(q_0,1) = q_3$

$\delta(q_0,0) = q_4$ | $\delta(q_4,1) = q_4$

marked | Not marked

q_0, q_4 are marked.

consider (q_1, q_5)

$\delta(q_1,0) = q_4$ | $\delta(q_1,1) = q_2$

$\delta(q_5,0) = q_2$ | $\delta(q_5,1) = q_4$

marked | marked

q_1, q_5 are marked.

consider q_3, q_4

$\delta(q_3,q_4) = q_0$ | $\delta(q_3,1) = q_4$

$\delta(q_4,0) = q_4$ | $\delta(q_4,1) = q_4$

marked | Not marked

q_3, q_4 are marked

consider (q_3, q_5)

$\delta(q_3,0) = q_0$ | $\delta(q_3,1) = q_4$

$\delta(q_5,0) = q_2$ | $\delta(q_5,1) = q_4$

marked | Not marked

q_3, q_5 are marked

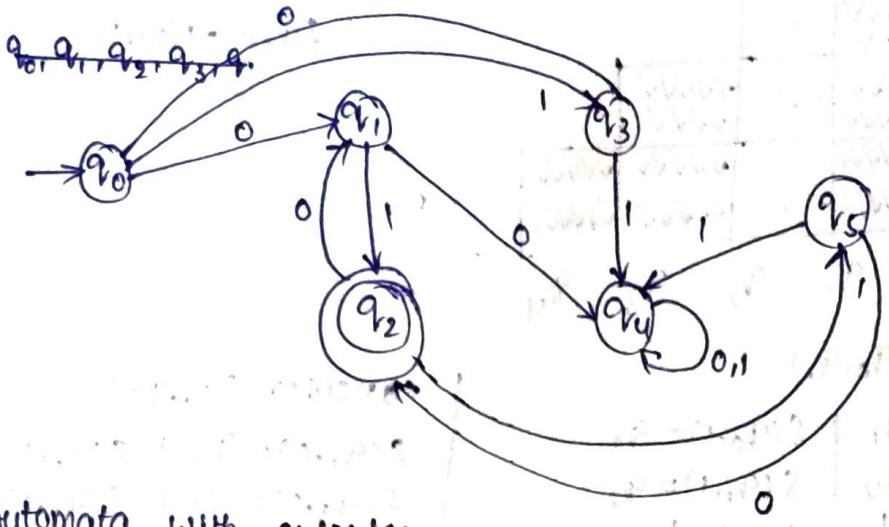
consider (q_4, q_5)

$\delta(q_4,0) = q_4$ | $\delta(q_4,1) = q_4$

$\delta(q_5,0) = q_2$ | $\delta(q_5,1) = q_4$

Marked | Not marked

q_4, q_5 are marked



*Finite Automata With outputs:

Finite Automata is a machine it produces output. Finite Automata with output divided into two types.

1. Mealy machine
2. Moore machine

1. Mealy Machine:

Mealy Machine was proposed by George mealy at Bell labs in 1960. Mealy machine is one type of finite Automata with output where output depends on present state & present input.

Mealy machine is Represented as $M = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$

Where Q: Finite non-empty set of states.

Σ: set of input Alphabets

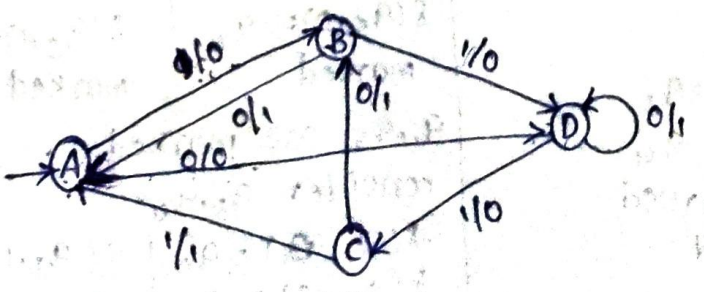
Δ: set of output Alphabets

δ: Transaction function mapping $Q \times \Sigma = Q$

λ: output function mapping $Q \times \Sigma = \Delta$

q_0: starting state.

Ex:



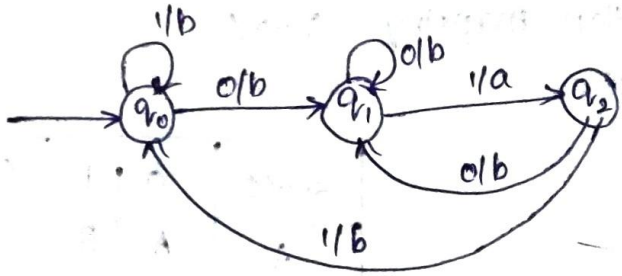
State	0	output	1	output
A	D	0	B	0
B	A	1	D	0
C	B	1	A	1
D	D	1	C	0

* construct mealy machine that produces 1's compliment of any Binary input string.

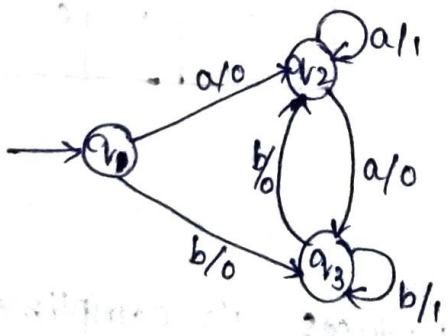
I/P	1	0
O/P	0	1

* construct mealy machine that prints a whenever the sequence of 0,1 is encountered at end in any input binary string.

$\Sigma = \{0,1\}$
$\Delta = \{a,b\}$



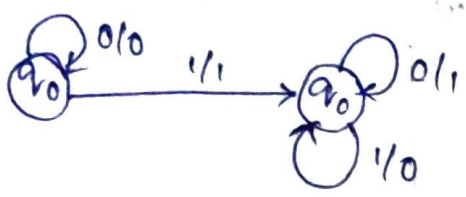
* construct mealy machine that accepts starting and ending with either aa or bb.



* construct mealy machine that accepts 2's compliment of any binary string.

111001111	1110 0	1111
000110000	0001 1	0000
+1	+1	+1
000110001	00100	0001

~~HHOO~~ ~~HH~~



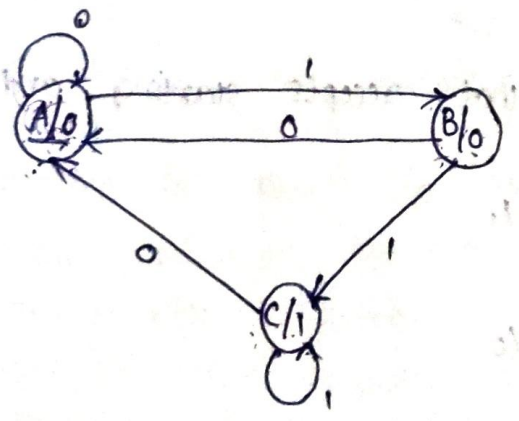
2. Moore Machine:

Moore machine was proposed by Edward F. Moore. in IBM around 1960's.

Moore machine is one type of Automata where output depends on present state only but the output is independent of present input.

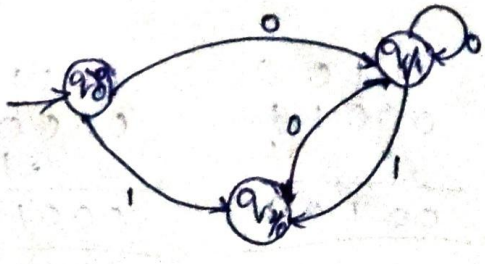
$$M = \{Q, \Sigma, \Delta, \delta, \xrightarrow{\lambda}, q_0\}$$

Where Q: Finite non empty set of states.
Σ: set of input Alphabets.
Δ: set of output Alphabets.
δ: Transition function mapping $Q \times \Sigma = Q$
λ: Output function mapping $Q \to \Delta$
q_0: starting state.

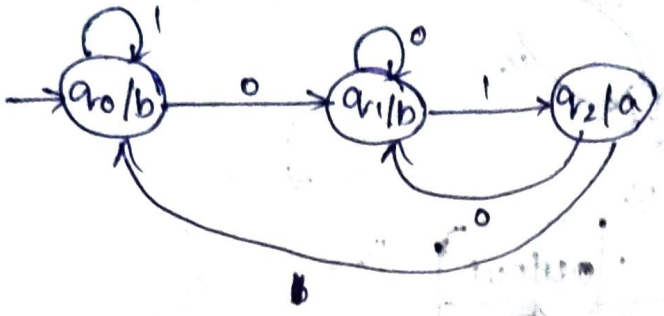


state	0	1	output
A	A	B	0
B	A	C	0
C	A	C	1

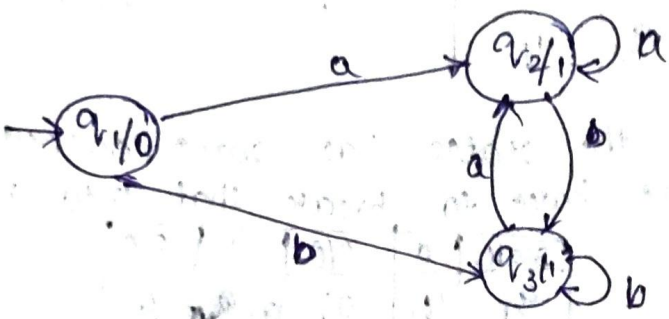
* construct Moore machine that produces 1's compliment for any binary string.



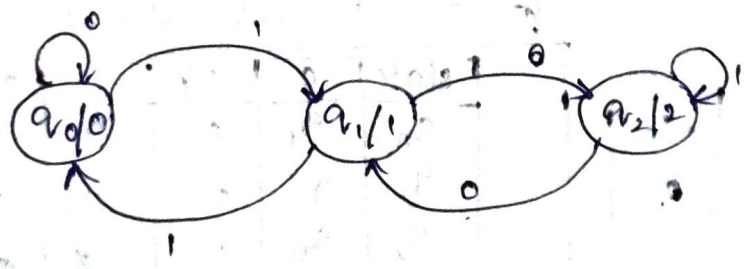
*construct Moore machine that prints a whenever the sequence 0,1 is encountered in any input binary string.



*construct moore machine that accepts string ending with aa, bb



* construct moore machine that takes binary number as input and produces ~~residue~~ remainder modulo 3.



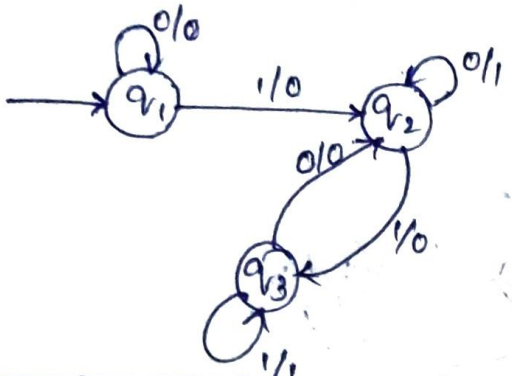
*Mealy to Moorey conversion:

let $M=\{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$ is Mealy machine then Moore machine is defined by $M'=\{Q', \Sigma, \Delta, \delta', \lambda', q_0\}$.

procedure:

1. Find no. of different outputs for each state.
2. If all outputs of state Q are same, copy the state Q in Moore machine. If it has n-distinct outputs, break state Q into n states like Q_1, Q_2, Q_3 ------ based on output.
3. If the output of state is 1, insert new state before this state in transition table.

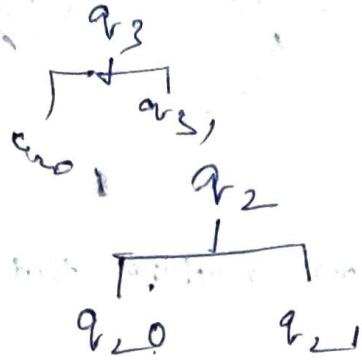
* convert the given mealy machine to moore machine.



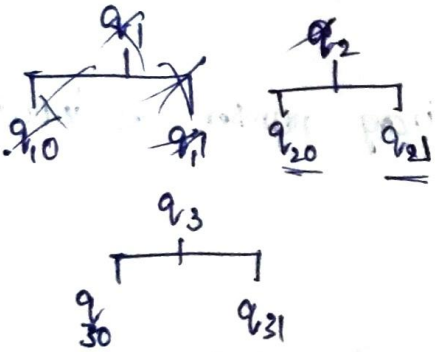
$\Sigma = \{0,1\}$

$\Delta = \{0,1\}$

state	0	output	1	output
q_1	q_1	0	q_2	0
q_2	q_2	1	q_3	0
q_3	q_2	0	q_3	1



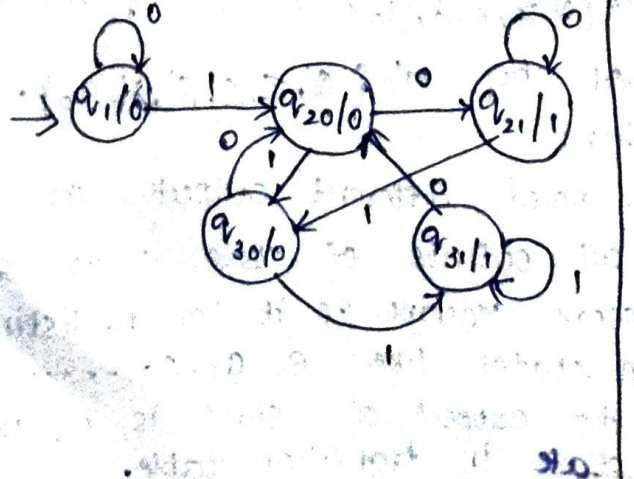
We have to see for the states has same output or not. If it is different we have to break that into number of different outputs.



state	0	1	o/p
q_1	q_1	q_{20}	0
q_{20}	q_{21}	q_{30}	0
q_{21}	q_{21}	q_{30}	1
q_{30}	q_{20}	q_{31}	0
q_{31}	q_{20}	q_{31}	1

state	0	o/p	1	o/p
q_1	q_{10}	0	q_{20}	0
q_{20}	q_{21}	1	q_{30}	0
q_{21}	q_{21}	1	q_{30}	0
q_{30}	q_{20}	0	q_{31}	1
q_{31}	q_{20}	0	q_{31}	1

state	0	o/p	1	o/p
q_0	q_1	0	q_2	0
q_1	q_1	0	q_3	1
q_2	q_4	1	∅	∅
q_3	q_4	1	q_2	0
q_4	q_1	0	q_3	1



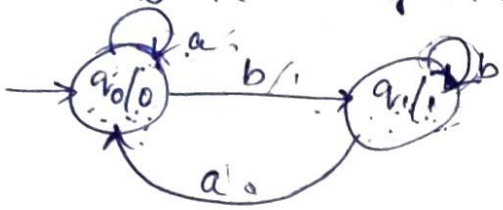
* Moore to Mealy conversion:

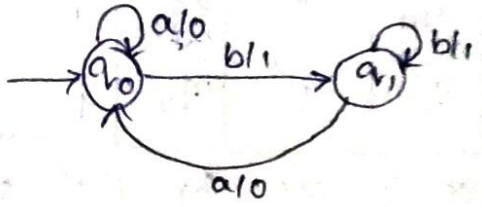
Let $M = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$ is mealy machine than moore machine is defined by $M' = \{Q, \Sigma, \Delta, \delta, \lambda', q_0\}$.

Procedure:-

For every state transactions of moore machine find the output by copying resultant state output in moore machine to the mealy machine.

*





State & output

q_0

q_1

State	a	o/p	b	o/p
q_0	q_0	0	q_1	1
q_1	q_0	0	q_1	1

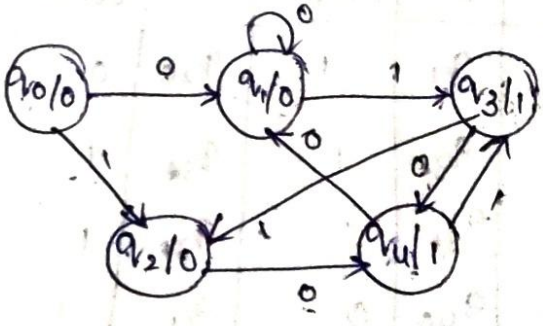
$\lambda'(q_0, a) = \lambda(\delta(q_0, a))$
$= \lambda(q_0)$
$= 0$

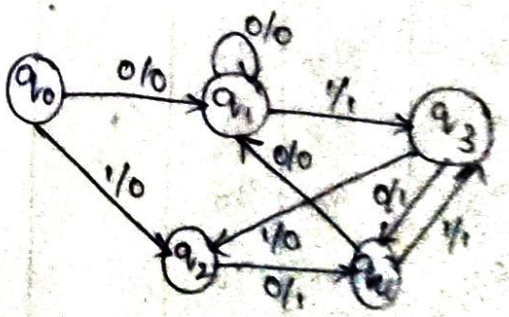
$\lambda'(q_0, b) = \lambda(\delta(q_0, b))$
$= \lambda(q_1)$
$= 1$

$\lambda'(q_1, a) = \lambda(\delta(q_1, a))$
$= \lambda(q_0)$
$= 0$

$\lambda'(q_1, b) = \lambda(\delta(q_1, b))$
$= \lambda(q_1)$
$= 1$

*





$\lambda'(q_0, 0) = \lambda(\delta(q_0, 0))$
$= \lambda(q_1)$
$= 0$

$\lambda'(q_0, 1) = \lambda(\delta(q_0, 1))$
$= \lambda(q_2)$
$= 0$

$\lambda'(q_1, 0) = \lambda(\delta(q_1, 0))$
$= \lambda(q_1)$
$= 0$

$\lambda'(q_1, 1) = \lambda(\delta(q_1, 1))$
$= \lambda(q_3)$
$= 1$

$\lambda'(q_2, 0) = \lambda(\delta(q_2, 0))$
$= \lambda(q_4)$
$= 1$

$\lambda'(q_2, 1) = \lambda((\delta(q_2, 1))$
$= \emptyset$

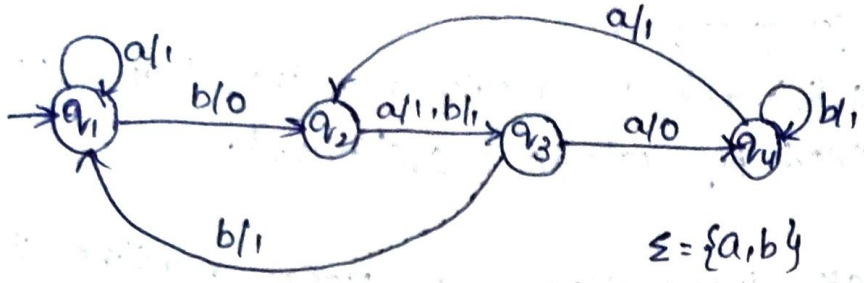
$\lambda'(q_3, 0) = \lambda(\delta(q_3, 0))$
$= \lambda(q_4) = 1$

$\lambda'(q_3, 1) = \lambda(\delta(q_3, 1)) = \lambda(q_4)$
$= 0$

$\lambda'(q_4, 0) = \lambda(\delta(q_4, 0))$
$= \lambda(q_1)$
$= 0$

$\lambda'(q_4, 1) = \lambda(\delta(q_4, 1))$
$= \lambda(q_3)$
$= 1$

*

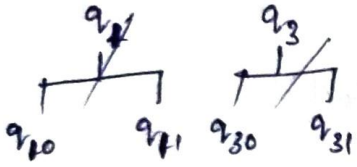


$\Sigma = \{a, b\}$

$\Delta = \{0, 1\}$

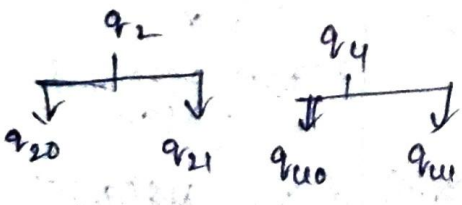
State	a	output	b	output
q_1	q_1	1	q_2	0
q_2	q_3	1	q_3	1
q_3	q_4	0	q_1	1
q_4	q_2	1	q_4	1

We have to see for the states has same output or not. If it is different we have to break that into no. of different outputs.



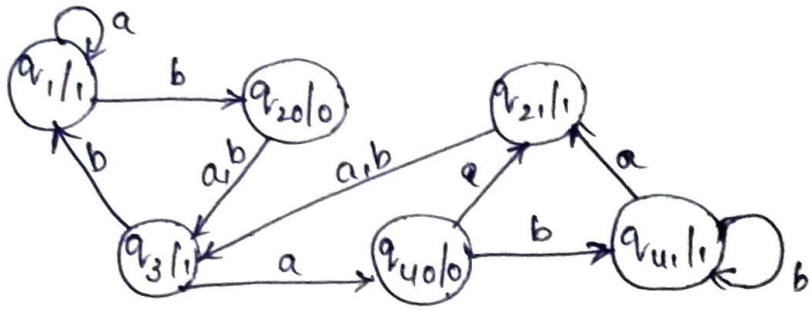
State	a	output	b	output
q_{10}	q_{11}		q_2	0
q_{11}	q_{10}		q_2	0
q_2	q_3	1	q_3	1
q_{30}	q_{40}	0	q_{11}	1
q_{31}	q_{40}	0	q_{11}	1
q_4	q_2	1	q_4	1

State	a	b	o/p
q_{10}	q_{10}	q_2	
q_{11}	q_{10}	q_2	
q_2	q_3	q_3	1
q_{30}	q_{40}	q_{11}	
q_{31}	q_{40}	q_{11}	
q_4	q_2	q_4	1



State	a	output	b	output
q_1	q_1	1	q_{20}	0
q_{20}	q_{30}	1	q_{30}	1
q_{21}	q_{30}	1	q_{30}	1
q_3	q_{40}	0	q_1	1
q_{40}	q_{21}	1	q_{41}	1
q_{41}	q_{21}	1	q_{41}	1

state	a	b	output
q_1	q_1	q_{20}	1
q_{20}	q_{30}	q_{30}	0
q_{21}	q_{30}	q_{30}	1
q_3	q_{40}	q_1	1
q_{40}	q_{21}	q_{41}	0
q_{41}	q_{21}	q_{41}	1



*convert residue modulo ~~file~~ 5 Moore machine to Mealy Machine.

$q_0/$

*Theorem:

If there is a NFA M, then there exists equivalence DFA M_1 that has equal string recognizing power.

Proof:

While discussing the proof we will concentrate on two things:

*HOW to construct the equivalent DFA? And

*the acceptability should be same for both.

Step1:

construction of the equivalent DFA M_1 from given NFA M

In NFA, zero, one or more next states are possible on a particular input. When we have more than one next state then we group all next states into one as $[q_1, q_2, q_3]$ and we call it one next state for equivalent DFA.

let $M = (Q, \Sigma, \delta, q_0, F)$ be the given NFA and $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be the equivalent DFA, then

1. $Q \subseteq 2^Q$ (2^Q is the power set of the set Q)
2. Σ is same for both.
3. $\delta = [q_0]$ is initial state for M_1.
4. $F_1 \subseteq 2^Q$ such that each member of F_1 has atleast one final state from F.
5. δ_1 is constructed as follows:

let $w = a \in \Sigma$ and

If for given NFA M: $\delta(q_0, a) = \{q_1, q_2, \dots, q_n\}$, then

For equivalent DFA $M_1 = \delta_1([q_0], a) = [q_1, q_2, \dots q_n]$

And

If for NFA M: $\delta(\{q_1, q_2 \dots q_n\}, a) = \{q_1, q_2 \dots, q_m\}$, then

For equivalent DFA M_1: $\delta_1([q_1, q_2 \dots q_n], a) = [q_1, q_2, \dots, q_m]$.

NOTE: $[q_1, q_2, \dots q_i]$ denotes a single state for equivalent DFA.

Step 2:

The acceptability of DFA and NFA.

We use the mathematical induction method to prove that $L(M) \subseteq L(M_1)$ and $L(M_1) \subseteq L(M)$ for all input strings $w \in \Sigma^*$

Case 1:

let $|w| = 0$, it means, $w = \epsilon$ (null string)

let w is accepted by NFA M if and only if

$\delta(q_0, \epsilon) = q_0$, and $q_0 \in F$ (starting state is final state)

so, the initial state of DFA will be the final state, hence $w = \epsilon$ is accepted by DFA also.

Case 2:

let $|w| = 1$ and $w = a \in \Sigma$ is accepted by NFA M, then for NFA $M: \delta(q_0, a) = \{q_1, q_2, \dots q_n\}$ and $\{q_1, q_2, \dots q_n\}$ has

at least one final state, then by constructive proof of equiva
lent DFA M_1:
$\delta_1([q_0], a) = [q_1, q_2, \dots q_n]$ and $[q_1, q_2 \dots q_n]$ has atleast one final state, so $[q_1, q_2, \dots, q_n]$ is a final state for equivalent DFA M_1.
Therefore, the equivalent DFA M_1 also accepts $w=a$.

case3:
suppose $|w|=n$ and $w = a_1 a_2 \dots a_n$ for accepted by both M & M_1 and for NFA $M: \delta'(q_0, a_1 a_2 \dots a_n) = \{q_1, q_2 \dots q_m\}$ and for equivalent DFA $M_1: \delta_1'([q_0], a_1 a_2 \dots a_n) = [q_1, q_2 \dots q_m]$

case4:
let $|w| = n+1$ and $w = yb$
where $|y| = n$, $y = a_1 a_2 \dots a_n$ and $y, b \in \Sigma^*$ is accepted by NFA M if and only if.
For NFA $M: \delta'(q_0, a_1 a_2 \dots a_n b) = \delta(\{q_1, q_2, \dots q_m\}, b) = \{q_1, q_2, \dots, q_p\}$. ($\{q_1, q_2, \dots q_p\}$ has atleast one final state from the set F).
By constructive proof of equivalent DFA M_1,
$\delta_1'([q_0], a_1, a_2 \dots a_n b) = \delta_1([q_1, q_2, \dots q_m], b) = [q_1, q_2, \dots q_p]$
$[q_1, q_2, \dots q_p]$ contains one final state from F, thus it is a final state equivalent DFA M_1.
Therefore, M_1 also accepts the string $w = yb$.
(δ', δ_1' are indirect transition functions for NFA M, and DFA M_1 respectively)
It has been proved that if NFA M accepts w then DFA M_1 also accepts w for any arbitrary string w.
Thus, $L(M_1) \subseteq L(M)$
Similarly, we can prove that if equivalent DFA M_1 accepts any string $w \in \Sigma^*$, then NFA also accepts it.
Thus, $M(L) \subseteq L(M_1)$

limitations of Finite Automata:
1. Finite automata can count Finite input.
2. There is no FA that can find and recognize set of binary string of equal 0's of 1's.
3. set of strings over '0' or '1' having balanced paranthesis.
4. Head movement is in only direction.

Applications of finite Automata:
1. FA is used in design of lexical analyzers.
2. Used for spell checkers.

3. Used for text editors.
4. Designing sequential circuits using mealy or moore machine
5. Recognizing patterns using regular expressions.

* DFA & NFA:

DFA is the short form for the deterministic finite Automata and NFA is for the Non-deterministic finite automata. NOW, let us understand in detail about these two finite Automata

DFA:

A Deterministic Finite Automata is a five-tuple automata. Following is the definition of DFA -

$M = (Q, \Sigma, \delta, q_0, F)$

where, Q: Finite set called states.
Σ: Finite set called alphabets.
$\delta: Q \times \Sigma \rightarrow Q$ is the transition function.
$q_0 \in Q$ is the start or initial state.
F: Final or accept state.

NFA:

NFA also has five states same as DFA, but with different transition function, as shown follows -

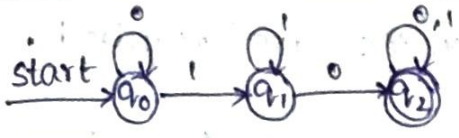
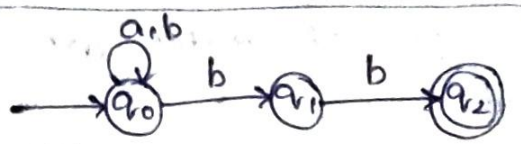
$\delta: Q \times \Sigma \rightarrow 2^Q$

where, Q: Finite set of states
Σ: Finite set of the input symbol.
q_0: initial state
F: Final state
δ: Transition Function.

Differences:

The major differences between the DFA and the NFA are as follows -

Deterministic Finite Automata.	Non-deterministic Finite Automata.
Each transition leads to exactly one state called as deterministic.	A transition leads to a sub set of states i.e. some transactions can be non-deterministic.
Accepts input if the last state is Final	Accepts input if one of the last states is in Final
Backtracking is allowed in DFA	Backtracking is not always possible.
Requires more space.	Requires less space.
Empty string transactions are not seen in DFA.	permits empty string transaction.

For a given state, on a given input we reach a deterministic and unique state.	For a given state, on a given input we reach more than one state.
DFA is a subset of NFA.	Need to convert NFA to DFA in the design of a compiler.
$\delta: Q \times \Sigma \rightarrow Q$. For example- $\delta(q_0, a) = \{q_1\}$	$\delta: Q \times \Sigma \rightarrow 2^Q$ For example- $\delta(q_0, a) = \{q_1, q_2\}$
DFA is more difficult to construct	NFA is easier to construct
DFA is understood as one machine.	NFA is understood as multiple small machines computing at the same time.
	

* Mealy Machine:

In a mealy machine the output symbol depends upon the present input symbol and present state of the machine.

In the mealy machine, the output is represented with each input symbol and each state is separated by /.

The Mealy machine can be described by six tuples $(Q, q_0, \Sigma, O, \delta, \lambda')$

Where, Q: Finite set of states

q_0: Initial state of machine

Σ: Finite set of input Alphabet

O: Output alphabet

δ: Transition Function where $Q \times \Sigma \rightarrow Q$

λ': output function where $Q \times \Sigma \rightarrow O$.

In the mealy machine, the output is represented with each input symbol and each state is seperated by /.

The length of output for a mealy machine is equal to the length of input.

Moore Machine:

Moore machine is a finite state machine. In which the next state is decided by the current state and current input symbol.

The output symbol at a given time depends only on the present state of the machine.

Moore machine has six tuples $(Q, q_0, \Sigma, O, \delta, \lambda)$

Where Q: Finite set of states.

q_0: Initial state of machine.

Σ: Finite set of input symbols.

O: Output Alphabet.

δ: Transition Function Where $Q \times \Sigma \rightarrow Q$.

λ: Output function Where $Q \rightarrow O$.

Differences:

The major differences between Mealy machine and Moore machine are as follows.

Mealy machine.	Moore machine.
Output depends on present state as well as present input.	Output depends only upon the present state.
If input changes, output also changes.	If input changes, output does not change.
less number of states are required.	More states are required.
Asynchronous output generation.	Synchronous output and state generation.
Output is placed on transition.	Output is placed on state
It is difficult to design.	Easy to design.