UNIT- I

- Why Study Automata Theory?
- The central Concept of Automata Theory
- Automation,
- Finite Automation
- Transition Systems
- Acceptance of a String by a finite Automation
- DFA
- Design of DFAs
- NFA
- Design of NFA
- Equivalence of DFA and NFA
- Conversion of NFA into DFA
- Finite Automata with ϵ-Transitions
- Minimization of finite Automata
- Mealy and Moore Machines
- Applications and Limitation of finite Automata.

The Central Concept of Automata Theory:

(•) Alphabet — A set of symbols
(•) Strings — A list of symbols from an
(•) Language. — A set of strings from the same alphabet.

(a) Alphabet: An alphabet is a collection of finite, nonempty set of symbols. It is denoted by Σ.

Ex: $\Sigma = \{0,1\}$ The binary alphabet

$\Sigma = \{a, b, \ldots z\}$, the set of all lower-case letters

$\Sigma = \{0, 1, 2, \ldots, 9\}$ Number

$\Sigma = \{0, 1, 2, \ldots, 9, a, b, \ldots, z, A, B, \ldots Z\}$ Alphanumeric Alphabets.

Powers of an Alphabet:

(b) String: (or Word (w))

A string is a finite sequence of symbols chosen from some alphabet Σ.

Ex: Consider binary alphabet $\Sigma = \{0,1\}$

Strings are $\{\in, 0, 1, 00, 01, 10, 11, \ldots\}$

Empty String:

The empty string is the string with zero occurrences of symbols. This string is denoted by ϵ.

Operations on Strings:

We can perform different operations on strings.

(1) length of a string
(2) Empty string
(3) prefix of string
(4) Suffix of string
(5) Concatenation of strings

length of a string:

The length of a string 'w' is no. of positions for symbols in the string. It is denoted by $|w|$. ie no. of symbols in string.

Eg: String $w = 01101$ length $|w| = 5$.

$|\epsilon| = 0$.

w = RISE, Then $|w| = 4$

w = Hello, Then $|w| = 5$.

Prefix and Suffix of String:

- Prefix of any string is any number of leading symbols of that string
- Suffix of any string is any number of trailing symbols of string.

Proper substring is any substring except string itself.

Ex: String = 'abc'

Prefix = ∈, a, ab, abc

Suffix : ∈, c, bc, abc

Substring : ∈, a, b, c, bc, ab, ~~abc~~ abc

Concatenation of Strings:

Let x and y be two string. Then xy denotes the concatenation of x and y. ie the string formed by making a copy of x and following it by a copy of y.

- If x is the string composed of i symbols $x = a_1 a_2 \cdots a_i$ and y is the string composed of j symbols $y = b_1 b_2 \cdots b_j$, then xy is the string of length i+j:

  ie $xy = a_1 a_2 \cdots a_i b_1 b_2 \cdots b_j$

Ex: x = 1101, y = 0011, then xy = 11010011.

yx = 00111101 . |x| = 4, |y| = 4, |xy| = 8, |yx| = 8.

for any string w, the equation ∈w = w∈ = w hold.

ie '∈' is the identity for concatenation.

Powers of an alphabet:

If Σ is an alphabet, the set of all strings of a certain length from that alphabet by using an exponential notation.

Σ^k be the set of strings of length k, each of whose symbols is in Σ.

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

Σ^0 = set of all strings of length zero (0) = $\Sigma^0 = \{\epsilon\}$

Σ^1 = set of all strings of length 1. $\Sigma^1 = \{0, 1\}$

Σ^2 = set of all strings of length 2, $\Sigma^2 = \{00, 01, 10, 11\}$

Σ^3 = set of all strings of length 3, $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$.

Σ^n = set of all strings of length n, $\Sigma^n =$ - -

Set of all strings over an alphabet Σ is conventionally denoted by Σ^*. ie $\{0,1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

ie $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$

Sometimes, we exclude the empty string from the set of strings. The set of nonempty strings from alphabet Σ is denoted by Σ^+.

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$$

$$\Sigma^* = \Sigma^+ \cup \{\epsilon\}$$

Languages:

A set of strings all of which are chosen from Σ^*, where Σ is a particular alphabet, is called a language.

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

Notice: a language over Σ need not include strings with all the symbols of Σ, so once we have established that L is a language over Σ, we also know it is a language over any alphabet that is a super set of Σ.

Ex: ⊙ The language of all strings consisting of n 0's followed by n 1's, for some $n \geq 0$:

$L = \{ \epsilon, 01, 0011, 000111, \cdots \}$

⊙ The set of strings of 0's and 1's with an equal number of each

$\{ \epsilon, 01, 10, 0011, 0101, 1001, 1100, \cdots \}$

⊙ The set of binary numbers whose value is a prime

$= \{ 10, 11, 101, 111, 1011, \cdots \}$

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

⊙ $\emptyset$, the empty language, is a language over any alphabet

⊙ $\{\epsilon\}$, the language consisting of only the empty string, it also a language over any alphabet. Notice that

$\emptyset \neq \{\epsilon\}$; $\emptyset$ has no strings
ϵ has one string.

Note:

* (i) $L \subseteq \Sigma^*$, L is language over Σ
  (ii) A language can be empty $L = \emptyset$
  (iii) Empty language is not equal to $L = \{\epsilon\}$
  ie $L \neq \{\epsilon\}$

Operations on language:

Various operations are performed on languages. They are (i) Union
(ii) Concatenation
(iii) closure properties.

(i) Union operation

(i) If L_1 and L_2 are two languages, then the Union is denoted by '$L_1 \cup L_2$' or '$L_1 + L_2$'.

$L_1 \cup L_2$ is a language that contain all strings from both L_1 and L_2. ie.

$$L_1 \cup L_2 = \{ w \mid w \text{ is in } L_1 \text{ or } w \text{ is in } L_2 \}.$$

Ex. (1.) $L_1 = \{0, 01, 10, 11\}$ $L_2 = \{0, 01, 001\}$ then

$L_1 \cup L_2 = \{0, 01, 10, 11, 001\}$.

(2) $L_1 = \{0, 00, 000, \ldots\}$ $L_2 = \{1, 11, 111, \ldots\}$

$L_1 \cup L_2 = \{0, 00, 000, \ldots 1, 11, 111, \ldots\}$

$L_1 = \{good, bad\}$, $L_2 = \{boy, girl\}$

$L_1 \cup L_2 = \{good, bad, boy, girl\}$.

(ii) Concatenation:

Concatenation of two languages is denoted by $L_1 L_2$.

$L_1 L_2 = \{uv \mid u \in L_1, v \in L_2\}$

Ex: ① $L_1 = \{good, bad\}$, $L_2 = \{boy, girl\}$

$L_1 L_2 = \{goodboy, goodgirl, badboy, badgirl\}$.

② let $\Sigma = \{0, 1\}$, A and B are 2 languages over Σ where $A = \{00, 11\}$, $B = \{0, 1\}$, find AB and BA.

Sol: $AB = \{000, 001, 110, 111\}$.

$BA = \{000, 011, 100, 111\}$.

③ $\Sigma = \{a, b\}$ where $A = \{ab, aa\}$, $B = \{bb, ba\}$. Find AB & BA

$AB = \{abbb, abba, aabb, aaba\}$

$BA = \{bbab, bbaa, baab, baaa\}$.

NOTE: ① $L_1 L_2 \neq L_2 L_1$.

② The Concatenation operation is Associative.

ie $L_1(L_2 L_3) = (L_1 L_2) L_3$

iii) Closure operations:

There are 2 types of closure operations. They are

(a) Kleen closure (Star closure)

(b) Positive closure

Kleen closure: Σ^*.

The language consisting of all words that are concatenation of 0 or more words in the original language (ie including null string ϕ also)

ie $\Sigma^* = \{\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \dots\}$

where $\Sigma^0 = \in$

Ex: $\Sigma = \{x\}$, $\Sigma^* = \{\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \dots\}$

$= \{\in, x, xx, xxx, \dots\}$

Ex: $S = \{01, 11\}$ find S^*

$S^* = \{\in, 01, 11, 0101, 1111, 0111, 1101, 010101, \dots\}$

Ex: $\Sigma = \{a, b\}$, $\Sigma^* = \{\in, a, ab, b, aab, bba, bb, \dots\}$

Positive closure:

The positive closure of a language is defined as

$$\Sigma^+ = \{\Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \dots\}$$

Eg:

$$\Sigma^+ = \Sigma^* - \{\in\}$$
$$\Sigma^* = \Sigma^+ + \{\in\}$$

Ex: Let $S = \{01, 11\} \Rightarrow S^+ = \{01, 11, 0111, 1101, 0101, 1111, 01111\dots\}$

2015

(3.b) Consider a language L^*, where $L = \{ab, cd\}$ with $\Sigma = \{a, b, c, d\}$

(i) write all words in L^* that have six or less letters/symbols.

(ii) what is the shortest string in Σ^* that is not in the language L^*?

Ans: The given language $L = \{ab, cd\}$

where alphabet $\Sigma = \{a, b, c, d\}$

(i) $L^* = \{\in, ab, cd, abcd, abab, cdcd, cdab, ababab, cdcdcd, abcdcd, cdabcd, cdcdab, cdabab, abcdab, ababcd \ldots\}$

The words in L^* that have six (or) less than six letters are :- $\in$, ab, cd, abcd, abab, cdcd, cdab, ababab, cdcdcd, abcdcd, cdabcd, cdcdab, cdabab, abcdab, ababcd.

(ii) Given $\Sigma = \{a, b, c, d\}$

$\Sigma^* = \in, a, b, c, d, ab, ac, ad, bc, bd, cd, aa, bb, cc, dd, ba, ca, da, cd, dc, \ldots$

Given $L = \{ab, cd\}$

$L^* = \{\in, ab, cd, abcd, abab, cdcd, cdab, ababab, cdcdcd, abcdcd, cdabcd, cdcdab, abcdab, cdabab, ababcd, \ldots\}$

∴ The shortest string in Σ^* that is not in L^* is

$w = \{a, b, c, d\}$

∴ The four strings having same length.

Ex: Consider the words $x = \{110\}$, $y = \{0110\}$.
Find (i) xy (ii) yx, (iii) x^2 (iv) ϵy (v) $x\epsilon$

Sol: Given $x = 110$, $y = 0110$

(i) $xy = 110\,0110$

(ii) $yx = 0110\,110$

(iii) $x^2 = xx = 110\,110$

(iv) $\epsilon y = \epsilon\,0110 = 0110$

(v) $x\epsilon = 110\epsilon = 110$

Ex: Describe in words the following over an alphabet $S = \{a, b\}$

(i) $L_1 = \{a, ab, ab^2, \cdots\}$

(ii) $L_2 = \{a^n b^n \mid n \geq 1\}$

(iii) $L_3 = \{a^m b^n \mid m \geq 0, n > 0\}$

Sol:

$S = \{a, b\}$

(i) $L_1 = \{a, ab, ab^2, \cdots\}$

$\therefore L_1$ is a language having strings starting with letter 'a' followed by zero(0) or many b's.

(ii) Let $L_2 = \{a^n b^n \mid n \geq 1\}$

$L_2 = \{ab, a^2b^2, a^3b^3, \cdots\} = \{ab, aabb, aaabbb, \cdots\}$

$\therefore L_2$ is a language with same number of a's and b's.

(iii) Let $L_3 = \{a^m b^n \mid m \geq 0, n > 0\}$

i.e $m = 0, 1, 2, 3, \cdots$, $n = 1, 2, 3, 4, \cdots$

$L_3 = \{b, ab, a^2b, a^3b, \cdots\} = \{b, abb, aabbb, aaabbbb, \cdots\}$

$\therefore L_3$ is a language.

Automation:

An automation is defined as a system where energy, materials and information are transformed, transmitted and used for performing some functions without direct participation of man.

Ex:- automatic machine tools,
automatic packing machines,
automatic photo printing machines etc.

In computer science the term 'automation' means 'discrete automation'

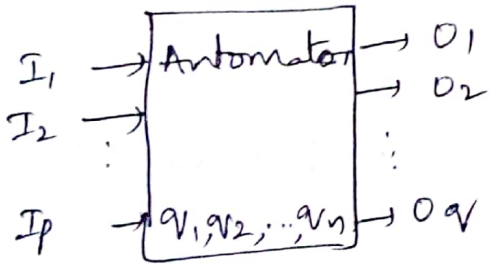


fig: model of discrete automation.

Characteristics of automata are

i) Input: At each of the discrete instants of time t_1, $t_2, \ldots t_m$ the input values $I_1, I_2, \ldots I_p$, each of which can take a finite number of fixed values from the input alphabet Σ, are applied to the input side of the model.

ii) Output: $O_1, O_2, \ldots O_q$ are the outputs of model, each of which can take a finite number of fixed values from an output O.

(iii) States: At any instant of time the automata can be in one of the states $q_1, q_2, \ldots q_n$.

(iv) State relation: The next state of an automaton at any instant of time is determined by the present state and present input.

(v) output Relation: The output is related to either state only or to both the input and the state. It should be noted that at any instant of time the automata is in some state. On 'reading' an input symbol, the automata moves to a next state which is given by the state relation.

Finite Automata : (FA)

A finite automata is formally defined as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

Q - is a finite set of states which is non-empty

Σ - is the input alphabet

q_0 - is the initial state

f - is the final ~~set of~~ states and $F \subseteq Q$

δ - is a transition function or mapping function

$Q \times \Sigma \to Q$ using this the next state can be determined depending on the current input.

Finite Automaton Model:

The finite automaton can be represented as input tape and finite control.

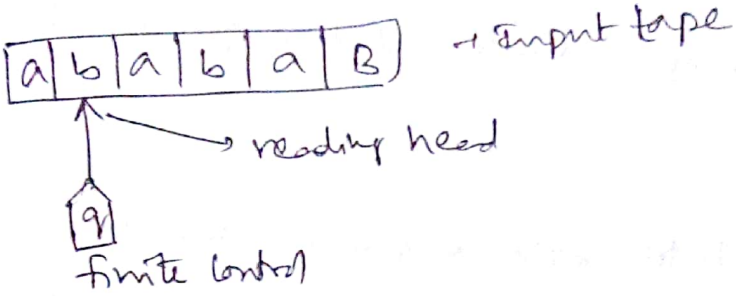


fig:- finite Automaton.

i) Input tape: Is a linear tape having some cells that can hold an input symbol from Σ.

ii) finite control: The finite control indicates the current state and it decides on the next state on receiving the particular input from input tape.

iii) Tape reader: The tape reader reads the cells one by one from left to right, and at any instance only one input symbol is read.

The reading head examines the read symbol, and the head moves to the right with or without changing the state. When the entire string is read, then it finite control is in final state, the string is accepted, else it is rejected.

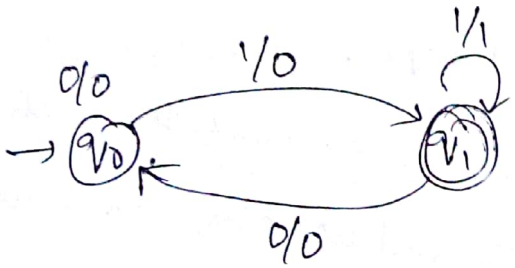
Transition Systems:

A transition graph or a transition system is a finite directed labelled graph in which each vertex (or node) represents a state and the directed edges indicate the transition of a state and edges are labelled with input/output.

In a typical transition system, the initial state is represented by a circle with an arrow pointing towards it, the final state by two concentric circles, and other states are represented by a circle.

The edges are labelled by input/output (eg 0/1, 1/0, 1/1, 0/0

Ex:-



If the system is in the state q_0 and the input 1 is applied, the system moves to state q_1 as there is a directed edge from q_0 to q_1 with label 1/0. It outputs 0.

Def:- A transition system is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$,

(i) Q, Σ and F are the finite nonempty set of states, the input alphabet, and the set of final states, respectively, as in the case of finite automata.

ii) $q_0 \subseteq Q$ and q_0 is nonempty; and

iii) δ is a finite subset of $Q \times \Sigma^* \times Q$.

In other words, if (q_1, w, q_2) is in δ, it means that the graph starts at the vertex q_1, goes along a set of edges, and reaches the vertex q_2. The concatenation of the label of all the edges thus encountered is w.

Def:- A transition system accepts a string w in Σ^* if

(i) there exists a path which originates from some initial state, goes along the arrows, and terminates at some final state, and

ii) The path value obtained by concatenation of all edge-labels of the path is equal to w.

properties of transition function

The transition function δ defined as $Q \times \Sigma \to Q$ holds the following properties for all states $q \in Q$ and $a \in \Sigma^*$

i) $\delta(q, \epsilon) = q$. The states of the FA are changed only by an input symbol.

ii) for all strings w and i/p symbol a.

$$\delta(q, aw) = \delta(\delta(q,a), w)$$

FA can be represented by a

(i) Transition diagram

(ii) Transition Table.

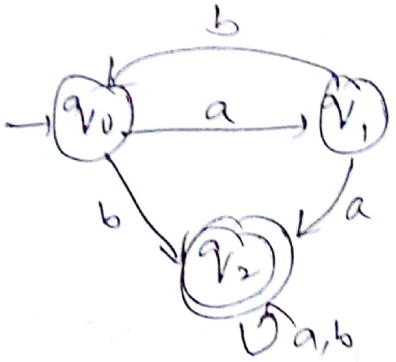
Transition graph contains

(i) set of states as circles

- Start state q_0 with arrow → q_0
- final state by double circle q_f

(ii) A finite set of transitions (edges/labels) that shows how to go from one state to another state.

Eg.



Transition diagram.

Transition Table

Following is a tabular representation where rows corresponds to states & columns corresponds to input. Start state is given by → and the final state by *.

$M = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$

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

for above Transition diagram,

$\delta(q_0, a) = q_1$ $\delta(q_2, a) = q_2$

$\delta(q_0, b) = q_2$ $\delta(q_2, b) = q_2$

$\delta(q_1, a) = q_2$

$\delta(q_1, b) = q_0$

δ/Σ	a	b
→q_0	q_1	q_2
q_1	q_2	q_0
*q_2	q_2	q_2

Acceptability of a String by a Finite Automata:

Def: A string x is accepted by a finite automata $M = \{Q, \Sigma, \delta, q_0, F\}$, if $\delta(q_0, x) = q$ for some $q \in F$.

This is basically the acceptability of string by the final state.

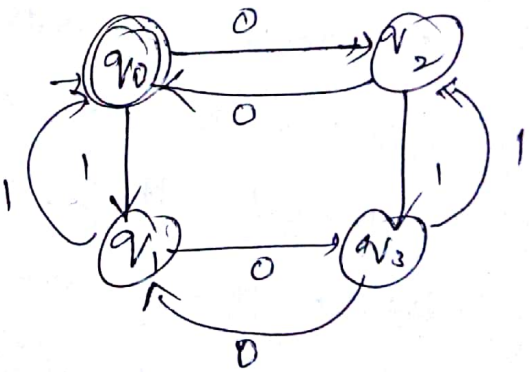
A final state is also called an accepting state.

Ex: Consider the finite state machine whose transition function δ is given in below table in the form of transition table. Here $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $F = \{q_0\}$. Give the entire sequence of states for the input string 1101001.

Transition function Table:

State	Input 0	1
→ q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Transition diagram



String: 1101001

$\delta(q_0, 1101001) \Rightarrow \delta(\delta(q_0, 1), 101001) = \delta(q_1, 101001) = \delta(\delta(q_1, 1), 01001)$

$= \delta(q_0, 01001) = \delta(\delta(q_0, 0), 1001) = \delta(q_2, 1001)$

$= \delta(\delta(q_2, 1), 001) = \delta(q_3, 001) = \delta(\delta(q_3, 0), 01)$

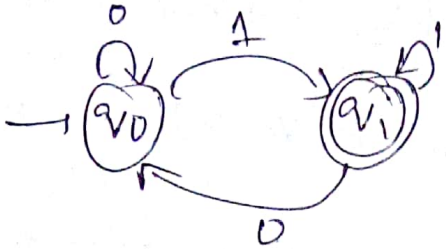
$= \delta(q_1, 01) = \delta(q_0, 1) = q_0$

1101001

$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_0$

q_0 is final state. String is accepted.

Ex:2 Give the language defined by the following F.A



Sol: different strings accepted by this automata are

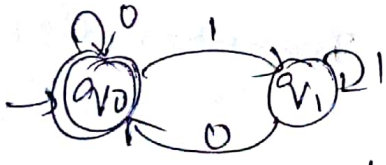
$\{1, 01, 001, 0001, 101, 1011, 1001, 10011, \ldots\}$

we observe that all the strings accepted are ending with 1.

$L(M) = \{w/w$ ends with 1 on $\Sigma = \{0,1\}\}$.

Language accepted by machine M (denoted by L(M)) is given by the set of strings that are ending with 1.

Ex: Identify the language defined by following machine.



Sol: In this automaton, since the initial state and the final state are same, it accepts ε and all strings that end with 0.

Hence the language accepted by this L(M).

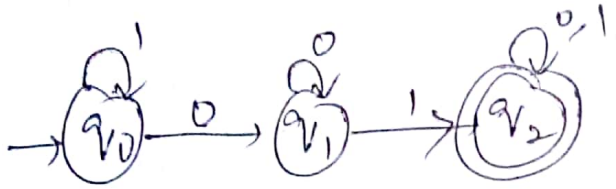
$L(M) = \{w/w$ is ε or all strings that end with $0\}$.

$L(\overline{M}) = \{w/w$ is ε or all strings do not end with $1\}$.

Ex:- Design a finite automata (or) transition diagram for accepting the string with substring 01. (10)

Sol:- Let $\Sigma = \{0, 1\}$

Language $L = \{01, 001, 010, 0100, 0010, \ldots\}$.



$M = \{Q, \Sigma, \delta, q_0, F\}$

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

$\Sigma = \{0, 1\}$ $\quad F = \{q_2\}$

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

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

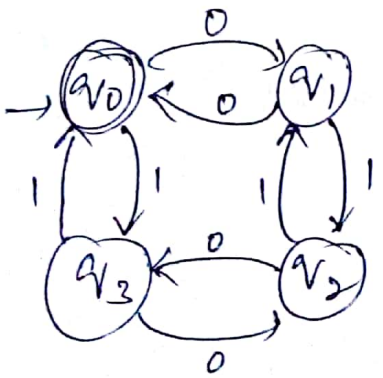
String 0010

$\delta(q_0, 0010) \vdash \delta(q_1, 010) \vdash \delta(q_1, 10) \vdash \delta(q_2, 0) \vdash q_2$

Ex:- Design FA which accepts even number of 0's and even number of 1's.

00, 011, 0101, 0011, 10110

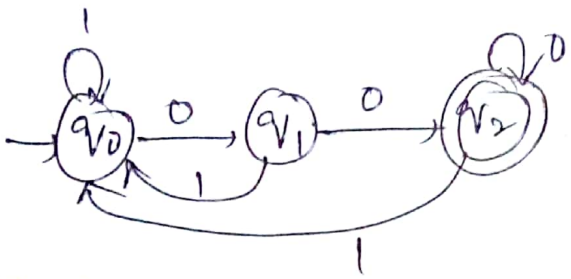
Sol.



	0	1
→ q_0	q_1	q_3
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_2	q_0

Ex: Design DFA to accept the string that always ends with 00. $\Sigma = \{0,1\}$

Sol: $\{00, 100, 1100, 10100, \cdots\}$



	0	1
→ q_0	q_1	q_0
q_1	q_2	q_0
(q_2)	q_2	q_0

$\Sigma = \{0,1\}$,
$Q = \{q_0, q_1, q_2\}$
$q_0 = \{q_0\}$
$F = \{q_2\}$

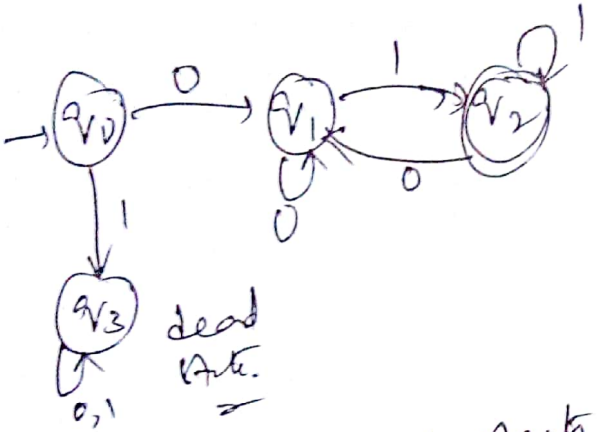
$\delta(q_0, 0) = q_1$
$\delta(q_0, 1) = q_0$
$\delta(q_1, 0) = q_2$
$\delta(q_1, 1) = q_0$
$\delta(q_2, 0) = q_2$
$\delta(q_2, 1) = q_0$

String: 01001100

$\delta(q_0, 01001100) \vdash \delta(q_1, 1001100)$
$\vdash \delta(q_0, 001100)$
$\vdash \delta(q_1, 01100)$
$\vdash \delta(q_2, 1100)$
$\vdash \delta(q_0, 100)$
$\vdash \delta(q_0, 00)$
$\vdash \delta(q_1, 0)$
$\vdash \{q_2\}$ string accepted

Ex: Construct the transition graph for a FA which accepts a language L over $\Sigma = \{0,1\}$ in which every string start with 0 and ends with 1.

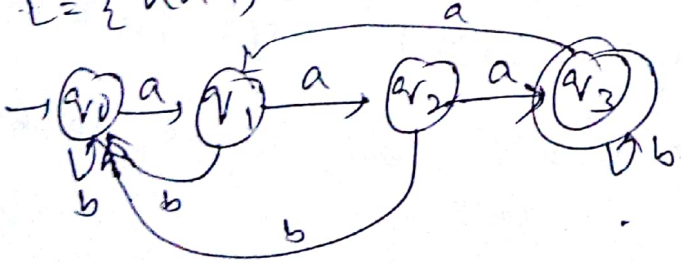
Sol: Strings: {01, 001, 011, 0101, 01101, ...}



Case: if the input starts with 1, then it will be in q_3 state, which is dead state and never lead to final state. Thus, this machine strictly handles the string starting with 0 and ending with 1.

Ex: Design FA to accept L, where L = {string in which a is always appears tripled} over the set $\Sigma = \{a,b\}$.

Sol: L = {aaa, aaab, baaa, bbaaa, babaaab ...}

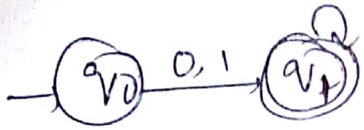


Ex:- Construct a finite automata that accepts {0,1}+

Sol:- Let $\Sigma = \{0,1\}$,

L = set of strings except ∈

$L = \{0, 1, 00, 01, 10, 11, \cdots\}$



$\delta(q_0, 0) = \{q_1\}$
$\delta(q_0, 1) = \{q_1\}$
$\delta(q_1, 0) = \{q_1\}$
$\delta(q_1, 1) = \{q_1\}$

Acceptance of string 0011

$\delta(q_0, 0011) \vdash \delta(\delta(q_0, 0), 011)$
$\vdash \delta(q_1, 011)$
$\vdash \delta(\delta(q_1, 0), 11)$
$\vdash \delta(\delta(q_1, 11)$
$\vdash \delta(\delta(q_1, 1), 1)$
$\vdash \delta(q_1, 1)$
$\vdash \{q_1\}$ final state - Accepted

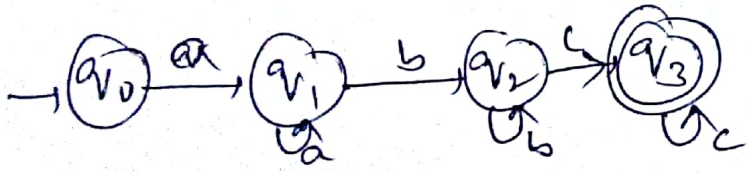
Ex:- $\{a^i b^j c^k \mid i, j, k > 0\}$

Sol: Let an alphabet, $\Sigma = \{a, b, c\}$

The language having strings that accepts $\{a^i b^j c^k \mid i, j, k > 0\}$

$L = \{abc, abcc, aabbcc, abbc, abbcc, aabbc, aabcc, \ldots\}$ etc

Since $i > 0$, ie $i = 1, 2, 3, \ldots$ etc
$j > 0$, ie $j = 1, 2, 3, \ldots$ etc
$k > 0$, ie $k = 1, 2, 3, \ldots$ etc.



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

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

$\delta : \delta(q_0, a) = \{q_1\}$

$\delta(q_1, a) = \{q_1\}$

$\delta(q_1, b) = \{q_2\}$

$\delta(q_2, b) = \{q_2\}$

$\delta(q_2, c) = \{q_3\}$

$\delta(q_3, c) = \{q_3\}$.

Acceptance of string:

Let the string $w = aabbcc$ accepts $\{a^i b^j c^k \mid i, j, k > 0\}$

$\delta(aabbcc) = \delta(\delta(q_0, a) abbcc)$

$= \delta(q_1, abbcc)$

$= \delta(\delta(q_1, a), bbcc)$

$= \delta(q_1, bbcc)$

$= \delta(\delta(q_1, b), bcc)$

$= \delta(q_1, bcc)$

$= \delta(\delta(q_1, b), cc)$

$= \delta(q_2, cc)$

$= \delta(\delta(q_2, c), c)$

$= \delta(q_2, c)$

$= \{q_3\}$ Accept final state.

Q1 Construct a finite automata that accepts $(0,1)^*$.

Sol Let alphabet = $\Sigma = \{0,1\}$

The Language having string accepts $\{0,1\}^*$ is

$L = \{\epsilon, 0, 1, 10, 11, 01, 110, 111, 000, 010, \ldots\}$

The required finite automata having strings that acc $\{0,1\}^*$ is shown in below figure

fig: finite automata accepts strings $\{0,1\}^*$

where $Q = \{q_0\}$
$\Sigma = \{0,1\}$
$q_0 = \{q_0\}$
$F = \{q_0\}$

δ: $\delta(q_0, 0) = \{q_0\}$
$\delta(q_0, 1) = \{q_0\}$

Sol: accept. $w = \{00011\}$

$\delta : \delta(q_0, 00011) \vdash \delta(\delta(q_0,0), 0011)$
$\vdash \delta(q_0, 0011)$
$\vdash \delta(\delta(q_0,0), 011)$
$\vdash \delta(q_0, 011)$
$\vdash \delta(\delta(q_0,0), 11)$
$\vdash \delta(q_0, 11)$
$\vdash \delta(\delta(q_0,1), 1)$
$\vdash \delta(q_0, 1)$
$\vdash \{q_0\}$ final state

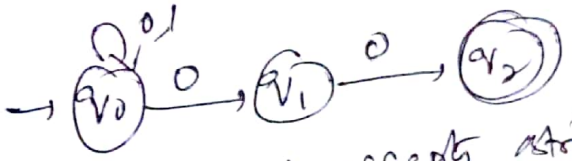
Q. Construct a finite automata that recognizes all possible strings over the alphabet $\Sigma = \{0,1\}$ ending with 2 consecutive zeros.

Sol. Let an alphabet $\Sigma = \{0,1\}$

$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, \ldots\}$

∴ The language L having strings ending with 2 consecutive zeros is: $L = \{00, 100, 1100, 000, 0100, \ldots\}$

∴ The required finite automata for the language 'L' is



Sol: F.A that accepts strings ending with 2 zeros.

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

$\Sigma = \{0,1\}$, $q_0 = \{q_0\}$, $F = \{q_2\}$

δ: $\delta(q_0,0) = \{q_0, q_1\}$ $\delta(q_1,0) = \{q_2\}$

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

Acceptance of string

Let $w = 1100$

$\delta: \delta(q_0, 1100) \vdash \delta(\hat{\delta}(q_0,1),100)$

$\vdash \delta(q_0, 100)$

$\vdash \delta(\hat{\delta}(q_0,1),00)$

$\vdash \delta(q_0, 00)$

$\vdash \delta(\hat{\delta}(q_0,0),0)$

$\vdash \delta(q_1, 0)$

$\vdash \{q_2\}$ final state.

So, the string '1100' accepted.

Q2) Draw a finite automata in which represents a blank.

Sol: Let $\Sigma = \{b, l, a, n, k\}$

The language L that accepts blank is $L = \{blank\}$

The required finite automata for language L is shown below

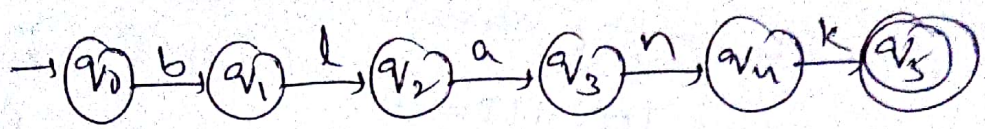


fig: FA represents a 'blank'

where $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

$\Sigma = \{b, l, a, n, k\}$, $q_0 = \{q_0\}$, $F = \{q_5\}$

δ: $\delta(q_0, b) = \{q_1\}$

$\delta(q_1, l) = \{q_2\}$

$\delta(q_2, a) = \{q_3\}$

$\delta(q_3, n) = \{q_4\}$

$\delta(q_4, k) = \{q_5\}$

Acceptance of string:

(i) let $w = blank$

$\delta: \delta(q_0, blank) \vdash \delta(\hat{\delta}(q_0, b), lank)$

$\vdash \delta(q_1, lank)$

$\vdash \delta(\hat{\delta}(q_1, l), ank)$

$\vdash \delta(q_2, ank)$

$\vdash \delta(\hat{\delta}(q_2, a), nk)$

$\vdash \delta(q_3, nk)$

$\vdash \delta(\hat{\delta}(q_3, n), k)$

$\vdash \delta(q_4, k)$

$\vdash \{q_5\}$ - final state

The string blank "blank" is accepted

Automata classification

The automata can be classified into two types

(1) Deterministic finite Automata (DFA)
(2) Non-deterministic finite Automata (NFA)

Deterministic finite Automata (DFA):

'Deterministic' refers to the fact that on each input there is one and only one state to which the automaton can transition from its current state.

In DFA, there will be a unique transition on any state on an input symbol.

DFA can be define as 5 tuple

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

where Q = finite set of states
Σ = finite set of input symbols
q_0 = initial state
F = final state
δ = transition function that takes two arguments, a state and an input symbol, and returns output as a state, i.e $\delta: Q \times \Sigma \to Q$.

Design DFA that accepts $M = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_0\})$ where δ is defined as the transition function

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

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

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

Sol:

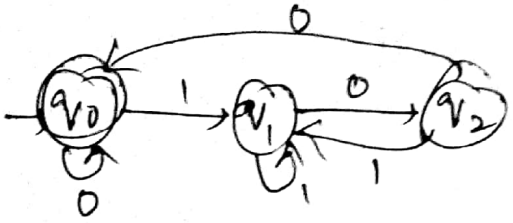


Fig: Required DFA

q_0 is initial & final state

Acceptance of string:

(i) $w = 010$

$\delta(q_0, 010) \vdash \delta(\hat{\delta}(q_0, 0), 10)$

$\vdash \delta(q_0, 10)$

$\vdash \delta(\hat{\delta}(q_0, 1), 0)$

$\vdash \delta(q_1, 0)$

$\vdash q_2$ not accepted.

(ii) $w = 100$

$\delta(q_0, 100) \vdash \delta(\hat{\delta}(q_0, 1), 00)$

$\vdash \delta(q_1, 00)$

$\vdash \delta(\hat{\delta}(q_1, 0), 0)$

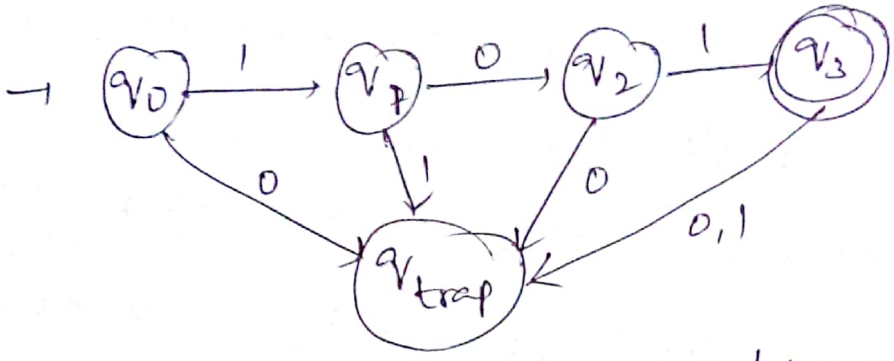
$\vdash \delta(q_2, 0)$

$\vdash q_0$ final state.

So, the string 100 is accepted

Ex: Design a DFA that accepts only input 101 over the set $\{0,1\}$.

Sol: $\Sigma = \{0,1\}$, $w = 101$



Here q_{trap} is called trap state / dummy state / dead state, where unnecessary transitions are thrown away.

$M = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{q_0, q_1, q_2, q_3, q_{trap}\}$

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

δ: $\delta(q_0,0) = \{q_{trap}\}$
$\delta(q_0,1) = \{q_1\}$
$\delta(q_1,0) = \{q_2\}$
$\delta(q_1,1) = \{q_{trap}\}$
$\delta(q_2,0) = \{q_{trap}\}$
$\delta(q_2,1) = \{q_3\}$
$\delta(q_3,0) = \{q_{trap}\}$
$\delta(q_3,1) = \{q_{trap}\}$

$w = 101$

$\delta(q_0,101) \vdash \delta(\hat{\delta}(q_0,1),01)$
$\vdash \delta(q_1,01)$
$\vdash \delta(\hat{\delta}(q_1,0),1)$
$\vdash \delta(q_2,1)$
$\vdash \{q_3\}$ Accepted.

Q1 Design DFA that accepts even number of 0's and even number of 1's.

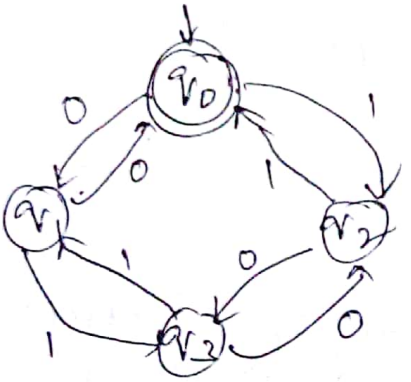
Ans

This DFA will have four different possibilities.

- Even number of 0's and even number of 1's - q_0
- Odd number of 0's and even number of 1's - q_1
- Even number of 0's and odd number of 1's - q_2
- Odd number of 0's and odd number of 1's - q_3

Where states are q_0, q_1, q_2 and q_3. Since the state q_0 indicates the condition of even number of 0's and even number of 1's. This state is made the final state.

The DFA is given by below fig.



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

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

$q_0 = \{q_0\}$

$F = \{q_0\}$

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

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

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

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

δ	0	1
→ q_0	q_1	q_2
q_1	q_0	q_3
q_2	q_3	q_0
q_3	q_2	q_1

$w = 1100$

$\delta: \delta(q_0, 1100) \vdash \delta(\hat{\delta}(q_0, 1), 100)$

$\vdash \delta(q_2, 100)$

$\vdash \delta(\hat{\delta}(q_2, 1), 00)$

$\vdash \delta(q_0, 00)$

$\vdash \delta(\hat{\delta}(q_0, 0), 0)$

$\vdash \delta(q_1, 0)$

$\vdash q_0$ final state. Accepted

w = 100

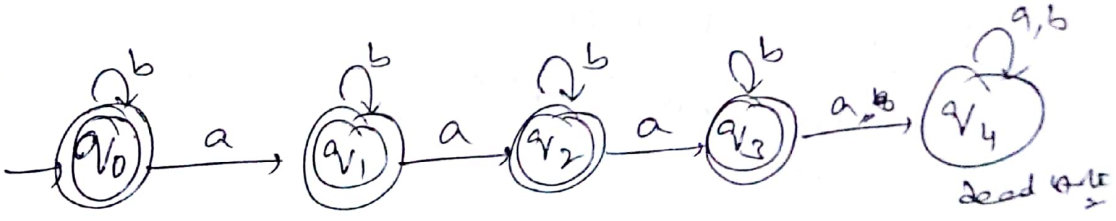
$\hat{\delta}: \hat{\delta}(q_0, 100) \vdash \hat{\delta}(\hat{\delta}(q_0, 1), 00)$
$\vdash \hat{\delta}(q_2, 00)$
$\vdash \hat{\delta}(\hat{\delta}(q_2, 0), 0)$
$\vdash \hat{\delta}(q_3, 0)$
$\vdash \{q_3\}$ not accepted, not final state

Ex: Design DFA that accepts all the strings with at most 3 a's.

Sol: Strings that have more than three a's should not be accepted. There are five possibilities. Hence, DFA requires 5 states.

No a's → q_0 (Accepting state)
one a → q_1 (Accepting state)
two a's → q_2 (Accepting state)
three a's → q_3 (Accepting state)
four a's → q_4 (for dummy state indicating a non-accepting state).

DFA:



$\Sigma = \{a, b\}$
$Q = \{q_0, q_1, q_2, q_3, q_4\}$
$q_0 = \{q_0\}$
$F = \{q_0, q_1, q_2, q_3\}$

$\delta(q_0, a) = \{q_1\}$, $\delta(q_0, b) = \{q_0\}$
$\delta(q_1, a) = \{q_2\}$, $\delta(q_1, b) = \{q_1\}$
$\delta(q_2, a) = \{q_3\}$, $\delta(q_2, b) = \{q_2\}$
$\delta(q_3, a) = \{q_4\}$, $\delta(q_3, b) = \{q_3\}$
$\delta(q_4, a) = \{q_4\}$, $\delta(q_4, b) = \{q_4\}$

	a	b
→ q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_2
q_3	q_4	q_3
q_4	q_4	q_4

w = bababaaab

$\delta(q_0, bababaaab)$

$\vdash \delta(\hat{\delta}(q_0, b), ababaaab)$

$\vdash \delta(q_0, ababaaab)$

$\vdash \delta(\hat{\delta}(q_0, a), babaaab)$

$\vdash \delta(q_1, babaaab)$

$\vdash \delta(\hat{\delta}(q_1, b), abaaab)$

$\vdash \delta(q_1, abaaab)$

$\vdash \delta(\hat{\delta}(q_1, a), baaab)$

$\vdash \delta(q_2, baaab)$

$\vdash \delta(\hat{\delta}(q_2, b), aaab)$

$\vdash \delta(q_2, aaab)$

$\vdash \delta(\hat{\delta}(q_2, a), ab)$

$\vdash \delta(q_3, ab)$

$\vdash \delta(\hat{\delta}(q_3, a), b)$

$\vdash \delta(q_4, b)$

$\vdash \{q_4\}$ not final state.

So, bababaaab NOT accepted

w = baa

$\delta(q_0, baa)$

$\vdash \delta(\hat{\delta}(q_0, b), aa)$

$\vdash \delta(q_0, aa)$

$\vdash \delta(\hat{\delta}(q_0, a), a)$

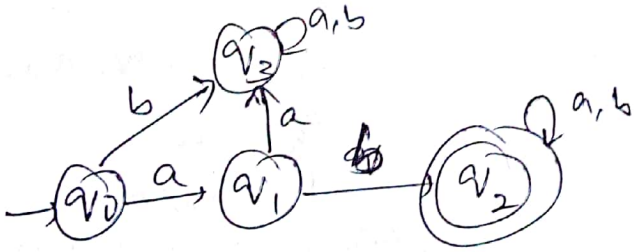
$\vdash \delta(q_1, a)$

$\vdash \{q_2\}$ final state

w accepted

Ex:- Draw a DFA that recognizes the set of all strings starting with prefix ab with alphabets from $\Sigma = \{a, b\}$

Sol:- This automaton should accept strings starts with ab. It should reject strings that starts with b. Hence if it sees b at the initial state, it should enter the dead state, from which there is no path to the final state. After 'a', it should find 'b'. Hence in the second state if it sees a, it should enter the dead state. Once it sees ab, it is the final state and remains in the same state on any element.



$Q = \{q_0, q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$q_0 = \{q_0\}$
$F = \{q_2\}$

$\delta(q_0, a) = \{q_1\}$ $\delta(q_1, a) = \{q_3\}$ $\delta(q_2, a) = \{q_2\}$ $\delta(q_3, a) = \{q_3\}$
$\delta(q_0, b) = \{q_3\}$ $\delta(q_1, b) = \{q_2\}$ $\delta(q_2, b) = \{q_2\}$ $\delta(q_3, b) = \{q_3\}$

	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_3	q_2
(q_2)	q_2	q_2
q_3	q_3	q_3

~~Accept~~
$w = aba$

$\hat{\delta}: \hat{\delta}(q_0, aba) \vdash \hat{\delta}(\hat{\delta}(q_0, a), ba)$
$\vdash \hat{\delta}(q_1, ba)$
$\vdash \hat{\delta}(\hat{\delta}(q_1, b), a)$
$\vdash \hat{\delta}(q_2, a)$
$\vdash \{q_2\}$
final state

'aba' Accepted

$w = bab$

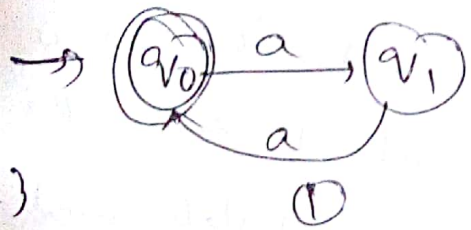
$\hat{\delta}: \hat{\delta}(q_0, bab)$
$\vdash \hat{\delta}(\hat{\delta}(q_0, b), ab)$
$\vdash \hat{\delta}(q_3, ab)$
$\vdash \hat{\delta}(\hat{\delta}(q_3, a), b)$
$\vdash \hat{\delta}(q_3, b)$
$\vdash \{q_3\}$ not final state

so, not accepted

Ex 1: Design DFA that accepts even number of a's over $\Sigma = \{a\}$.

Sol: This automaton should accept states that have even number of a's. On seeing a's which form an even count, the DFA should be in final state and on seeing odd count of a's it should be in non-final state.

DFA



(i)

$\Sigma = \{a\}$

$Q = \{q_0, q_1\}$

$q_0 = \{q_0\}$

$F = \{q_1\}$

$\delta(q_0, a) = \{q_1\}$

$\delta(q_1, a) = \{q_0\}$

Accept: w = {a}

$\delta: \delta(q_0, a) = \{q_1\}$ not final state

So, 'a' is not accepted.

w = aaaa

$\delta: \delta(q_0, aaaa) \vdash \delta(\delta(q_0, a), aaa)$

$\vdash \delta(q_1, aaa)$

$\vdash \delta(\delta(q_1, a), aa)$

$\vdash \delta(q_0, aa)$

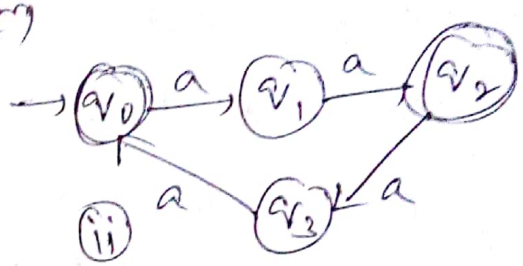
$\vdash \delta(\delta(q_0, a), a)$

$\vdash \delta(q_1, a)$

$\vdash \{q_0\}$ final state

Accepted

(ii)



$\Sigma = \{a\}$,

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

$q_0 = \{q_0\}$

$F = \{q_0, q_2\}$

$\delta(q_0, a) = \{q_1\}$ $\delta(q_2, a) = \{q_3\}$

$\delta(q_1, a) = \{q_2\}$ $\delta(q_3, a) = \{q_0\}$

w = aa

$\delta(q_0, aa) \vdash \delta(\delta(q_0, a), a)$

$\vdash \delta(q_1, a)$

$\vdash \{q_2\}$ final state

Accepted

w = aaa

$\delta(q_0, aaa) \vdash \delta(\delta(q_0, a), aa)$

$\vdash \delta(q_1, aa)$

$\vdash \delta(\delta(q_1, a), a)$

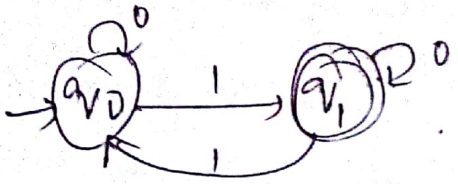
$\vdash \delta(q_2, a)$

$\vdash \{q_3\}$ not final state.

Not Accepted

Ex: DFA that accepts odd number of 1's on $\{0,1\}$

Sol:- This automaton should accept strings that have odd number of 1's and any number of 0's. On seeing 1's that form an odd count, the DFA should be in the final state and on seeing even count of 1's, it should be in the non-final state : 1, 010, 01101, 0101010 ..



$\Sigma = \{0,1\}$,
$Q = \{q_0, q_1\}$
$q_0 = \{q_0\}$
$F = \{q_1\}$

$\delta(q_0,0) = \{q_0\}$
$\delta(q_0,1) = \{q_1\}$
$\delta(q_1,0) = \{q_1\}$
$\delta(q_1,1) = \{q_0\}$

$w = 010$

$\delta(q_0, 010) \vdash \delta(\hat{\delta}(q_0,0),10)$
$\vdash \delta(q_0,10)$
$\vdash \delta(\hat{\delta}(q_0,1),0)$
$\vdash \delta(q_1,0)$
$\vdash \{q_1\}$ final state

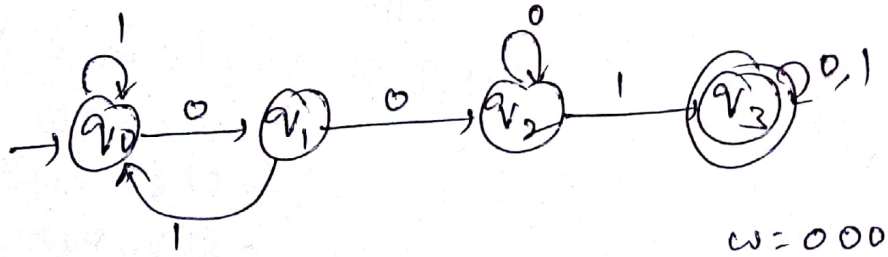
0110

$\delta(q_0,0110) \vdash \delta(\hat{\delta}(q_0,0),110)$
$\vdash \delta(q_0,110)$
$\vdash \delta(\hat{\delta}(q_0,1),10)$
$\vdash \delta(q_1,10)$
$\vdash \delta(\hat{\delta}(q_1,1),0)$
$\vdash \delta(q_0,0)$
$\vdash \{q_0\}$ not accepted

Ex: Design a DFA that contains 001 as substrings in all strings over $\Sigma = \{0,1\}$.

{ 001, 00010, 000110, 10010, 1001 ... }

Sol: The automaton should accept strings that have substring '001'. On seeing '0' go to next q_1 state, otherwise be in the same state. In q_1 state, on seeing '0', goto next q_2 state, otherwise go to initial state, as the substring would not be '00'. In q_2 state, on seeing '1', go to next q_3 state, otherwise be in the same state, as it would satisfy the required condition that 1 should be preceeded by '00'. Declare q_3 as final state.



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

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

$\delta(q_0, 0) = \{q_1\}, \delta(q_0, 1) = \{q_0\}$

$\delta(q_1, 0) = \{q_2\}, \delta\{q_1, 1\} = \{q_1\}$

$\delta(q_2, 0) = \{q_2\}, \delta\{q_2, 1\} = \{q_3\}$

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

$w = 0001$

$\hat{\delta}(q_0; 0001) \vdash \hat{\delta}(\delta(q_0, 0), 001)$

$\vdash \hat{\delta}(q_1, 001)$

$\vdash \hat{\delta}(\delta(q_1, 0), 01)$

$\vdash \hat{\delta}(q_2, 01)$

$\vdash \hat{\delta}(\delta(q_2, 0), 1)$

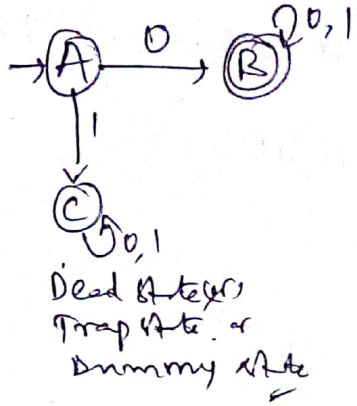
$\vdash \hat{\delta}(q_2, 1)$

$\vdash \{q_3\}$ final state

Accept

Ex:- Set of all strings that starts with '0'

Sol) $L = \{0, 00, 01, 000, 010, 011, 0000, \ldots\}$



Dead state / Trap state / Dummy state

$\Sigma = \{0,1\}$, $Q = \{A, B, C\}$

$q_0 = \{A\}$, $F = \{B\}$

$\delta(A,0) = \{B\}$ $\delta(B,0) = \{B\}$

$\delta(A,1) = \{C\}$ $\delta(B,1) = \{B\}$

$\delta(C,0) = \{C\}$

$\delta(C,1) = \{C\}$

$w = '010'$

$\delta: \delta(A, 010)$

$\vdash \delta(\hat{\delta}(A,0), 10)$

$\vdash \delta(B, 10)$

$\vdash \delta(\hat{\delta}(B,1), 0)$

$\vdash \delta(B, 0)$

$\vdash \{B\}$ final state

Accepted

$w = 10$

$\delta(A, 10) \vdash \delta(\hat{\delta}(A,1), 0)$

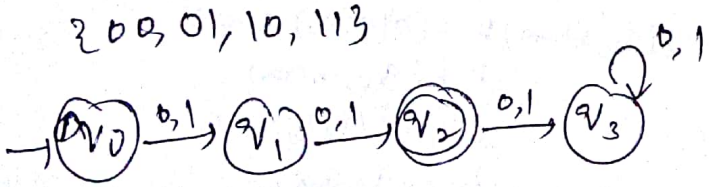
$\vdash \delta(C, 0)$

$\vdash \{C\}$ not a final state

'10' not Accepted.

Ex:- Construct a DFA that accepts set of all strings over $\{0,1\}$ of length 2.

Sol) $\{00, 01, 10, 11\}$



$\Sigma = \{0,1\}$, $Q = \{q_0, q_1, q_2, q_3\}$

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

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

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

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

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

$\delta(q_2,0) = \{q_3\}$

$\delta(q_2,1) = \{q_3\}$

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

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

$w = 01$

$\delta(q_0, 01) \vdash \delta(\hat{\delta}(q_0,0), 1)$

$\vdash \delta(q_1, 1)$

$\vdash \{q_2\}$ Accepted

final state

$w = 011$

$\delta(q_0, 011) \vdash \delta(\hat{\delta}(q_0,0), 11)$

$\vdash \delta(q_1, 11)$

$\vdash \delta(\hat{\delta}(q_1,1), 1)$

$\vdash \delta(q_2, 1)$

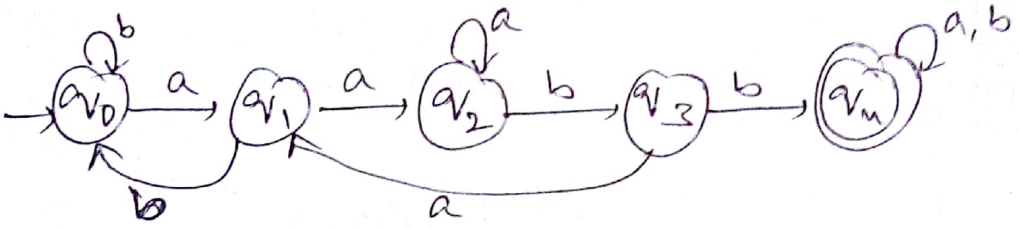
$\vdash \{q_3\}$ not final state

Not Accepted

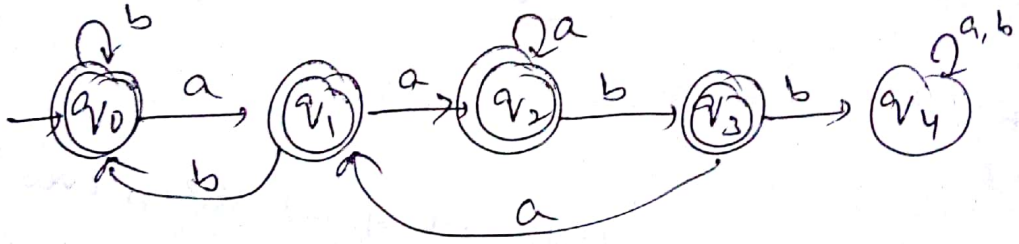
Ex:- Construct DFA that accepts any strings over {a,b} that does not contain the string aabb in it.

Sol:- $\Sigma = \{a, b\}$

- Let us contain a DFA that contain all strings over {a,b} that contain the substring aabb in it.



- Does not contain string aabb in it, then make non-final states to final state and final to to non-final state.



$\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3, q_4\}$

$q_0 = \{q_0\}$

$F = \{q_0, q_1, q_2, q_3\}$ ∀.

$\delta(q_0, a) = \{q_1\}$
$\delta(q_0, b) = \{q_0\}$
$\delta(q_1, a) = \{q_2\}$
$\delta(q_1, b) = \{q_1\}$
$\delta(q_2, a) = \{q_2\}$
$\delta(q_2, b) = \{q_3\}$
$\delta(q_3, a) = \{q_1\}$
$\delta(q_3, b) = \{q_4\}$
$\delta(q_4, a) = \{q_4\}$
$\delta(q_4, b) = \{q_4\}$

bbaa ✓

$\delta(q_0, bbaa) \vdash \delta(\hat{\delta}(q_0, b), baa)$
$\vdash \delta(q_0, baa)$
$\vdash \delta(\hat{\delta}(q_0, b), aa)$
$\vdash \delta(q_0, aa)$
$\vdash \delta(\hat{\delta}(q_0, a), a)$
$\vdash \delta(q_1, a)$
$\vdash \{q_1\}$ final state ∴ Accepted

aabb ✗

$\delta(q_0, aabb) \vdash \delta(\hat{\delta}(q_0, a), abb)$
$\vdash \delta(q_1, abb)$
$\vdash \delta(\hat{\delta}(q_1, a), bb)$
$\vdash \delta(q_2, bb)$
$\vdash \delta(\hat{\delta}(q_2, b), b)$
$\vdash \delta(q_3, b)$
$\vdash \{q_4\}$ non final state, ∴ not accepted

DFA Deterministic finite Automata

Def: A deterministic finite automata is a finite state machine where for each pair of states and input symbol, there is a unique next state.

Elements of DFA:

The deterministic finite automata exhibits the following five characteristics.

(i) a finite set of states Q.
(ii) an alphabet Σ of possible input symbols
(iii) a transition function $\delta : Q \times \Sigma \to Q$
(iv) the initial state $q_0 \in Q$
(v) the set of final states (F), where $F \subseteq Q$.

formally, a DFA, M is a five tuple

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

Design of DFA's

The basic design strategy for DFA is as follows

(i) Understand the language properties for which the DFA has to be designed.
(ii) Determined the state set required.
(iii) Identify the initial, accepting and dead state of DFA.
(iv) For each state, decide on the transition to be made for each character of the input string.
(v) Obtain the transition table and diagrams for DFA.
(vi) test the DFA obtained on short strings.

To design a DFA that accepts set of all strings that contain 0s or 1's and ends with 00.

Sol:- We are required to design a DFA for the regular expression $r = (0+1)^*00$ ... . In other words the DFA should be designed to accept the language of r.

i.e. $L(M) = \{00, 100, 1100, 000, 1000, 111000, 11100 \ldots\}$

where M is a DFA.

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

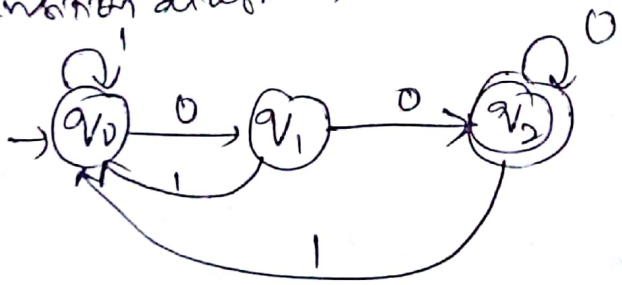
$Q = \{q_0, q_1, q_2\}$

q_0 = initial state

q_2 = final state

$\delta : Q \times \Sigma \to Q$

transition diagram



Q \ Σ	Input	
	0	1
→ q_0	$\{q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_0\}$
* q_2	$\{q_2\}$	$\{q_0\}$

Input

DFA action for the input string

i) To show that the string 100 is accepted by DFA:

ii) Using sequence diagram.

Since we encountered the end of the input and we are in the final state, we say that the string is accepted by machine M. Thus 100 is in L(M).

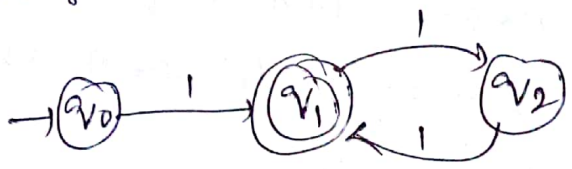
Ex: To design DFA that accepts odd number of 1's.

Sol: We are required to design a DFA, M for the regular expression $r = (11)^*1$ or $r = 1(11)^*$. In other words, DFA should be designed to accept the language of r.

i.e $L(M) = \{1, 111, 11111, 1111111, \ldots\}$

$\Sigma = \{1\}$, $Q = \{q_0, q_1, q_2\}$, q_0 = initial state, q_1 = final state.

δ is given by transition diagram



Ex: Transition diagram for $L(M) = \{w \in \Sigma^* \mid w$ is odd number of 1's$\}$

Transition table

δ / Σ	Input 1
→ q_0	q_1
* q_1	q_2
q_2	q_1

String 111

$\delta(q_0, 111) \vdash \delta(\delta(q_0,1),11)$
$\vdash \delta(q_1, 11)$
$\vdash \delta(\delta(q_1,1),1)$
$\vdash \delta(q_2, 1)$
$\vdash \{q_1\}$ final state
= Accept

String 11

$\delta(q_0, 11) \vdash \delta(\delta(q_0,1),1)$
$\vdash \delta(q_1, 1)$
$\vdash \{q_2\}$.
not final state, so not accepted

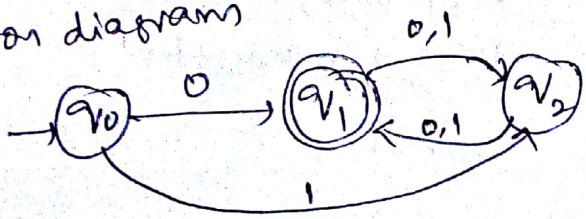
Ex: To design a DFA that
(i) start with 0 and has odd number of 0's.
(ii) starts with 1 and has even number of 1's.

Sol: We required to design a DFA for

$L(M) = \{01100, 11001, 0, 000, 11, \ldots\}$

$\Sigma = \{0,1\}$, $Q = \{q_0, q_1, q_2\}$. q_0 = initial state, q_1 = final state

δ is given by
Transition diagram



- Using extended transition function.

$$\delta(q_0, 100) = \delta(\delta(q_0, 1), 00)$$
$$= \delta(q_0, 00)$$
$$= \delta(\delta(q_0, 0), 0)$$
$$= \delta(q_1, 0)$$
$$= \{q_2\} \text{ final state}$$

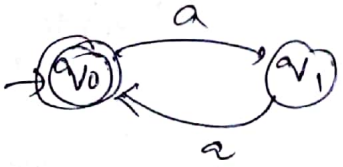
Ex: To design a DFA that accepts a set of even number of a's

Sol: We are required to design a DFA, M for the regular expression $r = (aa)^*$. In other words, DFA should be designed to accept the language of r.

i.e $L(M) = \{aa, aaaa, aaaaaa, \ldots\}$

Consider: $\Sigma = \{a\}$, $Q = \{q_0, q_1\}$, q_0 = initial state, q_1 = final state

& is given by transition diagram



$L(M) = \{w \in \Sigma^* \mid w \text{ is even}\}$

	a
→* q_0	q_1
q_1	q_0

String : aaaa

$\delta(q_0, aaaa) \vdash \delta(\delta(q_0, a), aaa)$
$\vdash \delta(q_1, aaa)$
$\vdash \delta(\delta(q_1, a), aa)$
$\vdash \delta(q_0, aa)$
$\vdash \delta(\delta(q_0, a), a)$
$\vdash \delta(q_1, a)$
$\vdash \{q_0\}$ final state
- Accepted

$w = aaa$

$\delta(q_0, aaa) \vdash \delta(\delta(q_0, a), aa)$
$\vdash \delta(q_1, aa)$
$\vdash \delta(\delta(q_1, a), a)$
$\vdash \delta(q_0, a)$
$\vdash \{q_1\}$ not a final state
not accepted

Transition Table

Σ / Input	0	1
→ q_0	q_1	q_2
* q_1	q_2	q_2
q_2	q_1	q_1

String w = 11001

$\delta(q_0, 11001) =$
$\delta(\delta(q_0, 1), 1001)$
$\vdash \delta(q_2, 1001)$
$\vdash \delta(\delta(q_2, 1), 001)$
$\vdash \delta(q_1, 001)$
$\vdash \delta(\delta(q_1, 0), 01)$
$\vdash \delta(q_2, 01)$
$\vdash \delta(\delta(q_2, 0), 1)$
$\vdash \delta(q_1, 1)$
$\vdash \delta(\delta(q_1, 1), \lambda)$
$\vdash \delta(q_2, \lambda)$
$\vdash \{q_1\}$ final state
$=$ Accepts

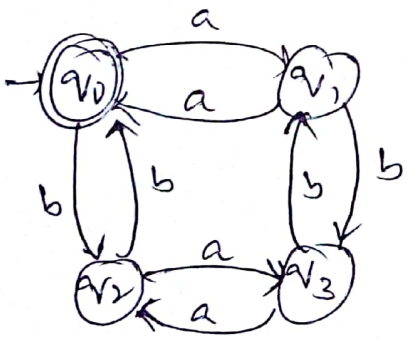
w = 01100 (1)

$\delta(q_0, 01100) \vdash \delta(\delta(q_0, 0), 1100)$
$\vdash \delta(q_1, 1100)$
$\vdash \delta(\delta(q_1, 1), 100)$
$\vdash \delta(q_2, 100)$
$\vdash \delta(\delta(q_2, 1), 00)$
$\vdash \delta(q_1, 00)$
$\vdash \delta(\delta(q_1, 0), 0)$
$\vdash \delta(q_2, 0)$
$\vdash \{q_1\}$ final state
$=$ Accepts

Ex:- To design a DFA to accept even number of a's and b's.

Sol:- This is to design a DFA, M for the

$L(M) = \{aa, bb, abab, baba, aabb, bbaa, \ldots\}$



Σ / Input symbols	a	b
→ * q_0	q_1	q_2
q_1	q_0	q_3
q_2	q_3	q_0
q_3	q_2	q_1

Transition Table

Fig: Transition diagram for $L(M) = \{w \in \Sigma^* \mid$ w has even length of a's and b's$\}$

abab

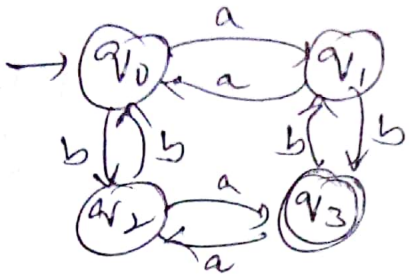
$\delta(q_0, abab) \vdash \delta(\delta(q_0, a), bab)$
$\vdash \delta(q_1, bab)$
$\vdash \delta(\delta(q_1, b), ab)$
$\vdash \delta(q_3, ab)$
$\vdash \delta(\delta(q_3, a), b)$
$\vdash \delta(q_2, b)$
$\vdash \{q_0\}$ Accepts

Ex: Design a DFA to accept odd number of a's and odd number of b's.

Sol: $L(M) = \{ab, ba, aaabbb, abbaba, \ldots\}$

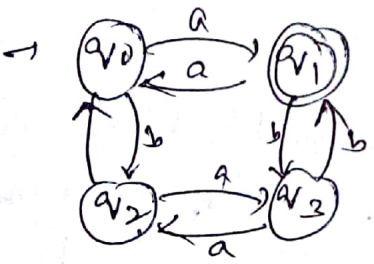
$\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3\}$, q_0 = initial state, q_3 = final state.

δ is given by

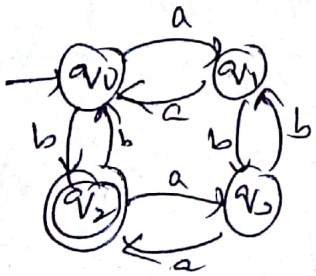


$\rightarrow q_0$

Design a DFA to accept odd number of a's and even no. of b's



Design a DFA to accept even number of a's and odd number of b's



NFA : Non-Deterministic Finite Automata

In this automaton, for a given input symbol, there can be more than one transition from a state. Such automaton is called Non-Deterministic finite Automata.

Non-deterministic finite automata can be defined as

$M = \{Q, \Sigma, \delta, q_0, F\}$

Where Q = Non-empty finite set of states

Σ = input alphabets

q_0 = initial start state

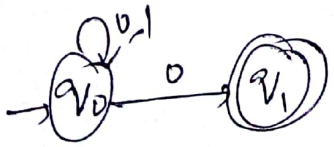
F = set of final state,

δ = transition function that takes two arguments (a state and an input symbol) and returns an output state, i.e

$\delta : Q \times \Sigma \to 2^Q$.

Ex:- Design NFA, which accepts the strings that ends with 0.

Sol) Strings: 0, 00, 100, 10, 1100, 110, ...



$M = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{q_0, q_1\}$

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

$q_0 = \{q_0\}$

$F = \{q_1\}$

	0	1
→ q_0	$\{q_0, q_1\}$	q_0
q_1	$\emptyset$	$\emptyset$

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

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

$\delta(q_1, 0) = \{\emptyset\}$

$\delta(q_1, 1) = \{\emptyset\}$

$\delta = Q \times \Sigma \to 2^Q$

$= 2 \times 2 = 2^2 = 4$

$\delta(q_0, 100)$

$\vdash \delta(q_0, 1) = \{q_0\}$

$\vdash \delta(\delta(q_0, 1), 0) =$

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

$\delta(q_0, 100) = \delta(\{q_0, 10\}, 0)$

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

$\vdash \{q_0, q_1\} \cup \{\emptyset\}$

$\vdash \{q_0, q_1\}$.

Let $P = \{q_0, q_1\}$, since P containing q_1, which is final state, so 100 is accepted by L(M)

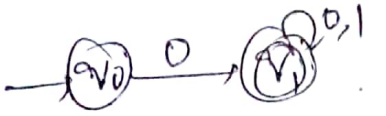
String 100.

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

- $q_0 \xrightarrow{1} q_0 \xrightarrow{0}$ $q_0 \xleftarrow{0} q_0$ (q_1)
- $(q_1) \rightarrow \emptyset$

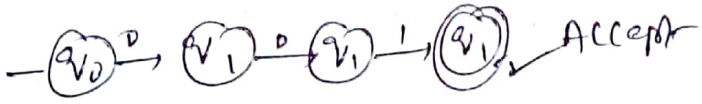
Ex:- Design NFA that accepts all the strings that start with '0'.

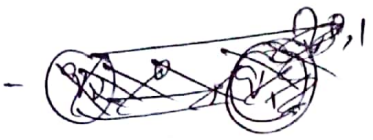
Sol:- $L = \{0, 00, 01, 000, 001, \ldots\}$



	0	1
q_0	q_1	$\emptyset$
q_1	q_1	q_1

String 001,





$M = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{q_0, q_1\}$

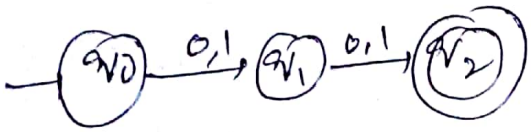
$\Sigma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_1\}$

Ex:- Construct NFA that accepts set of all strings over $\{0,1\}$ of length 2.

Sol:- $\Sigma = \{0,1\}$, $L = \{00, 01, 10, 11\}$



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

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

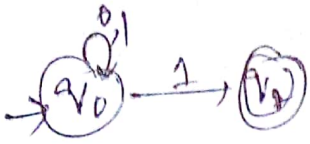
$q_0 = \{q_0\}$

$f = \{q_2\}$

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

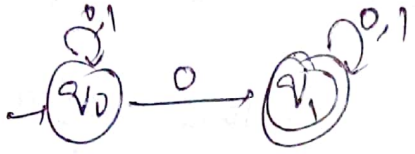
Ex: NFA, Set of all strings that ends with 1.

Sol: 1, 01, 11, 001, 111, 011, ... 0*1 ...



	0	1
→ q_0	$\{q_0\}$	$\{q_0, q_1\}$
(q_1)	$\emptyset$	$\emptyset$

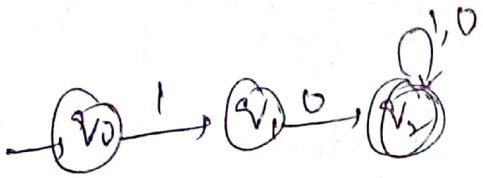
Ex: Set of all strings that contain '0'



0, 01, 101, 000, 100, ...

Ex: Set of all strings that starts with '10'

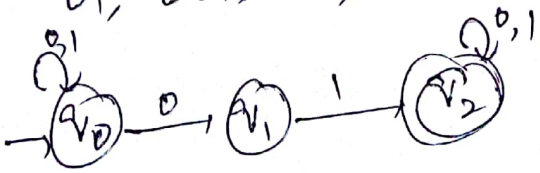
10, 101, 100, 1010, 1000, ...



	0	1
→ q_0	$\emptyset$	$\{q_1\}$
q_1	$\{q_2\}$	$\emptyset$
(q_2)	$\{q_2\}$	$\{q_2\}$

Ex: Set of all strings that contain '01'

Sol: 01, 001, 101, 0010, 0011, 1011 ...



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

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

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

$q_0 = \{q_0\}$

$F = \{q_2\}$

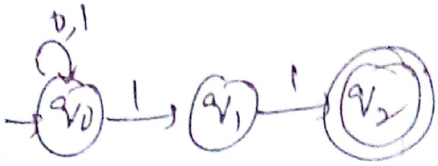
$\delta : Q \times \Sigma \to 2^Q$

$2^3 = 8$

Ex: set of all strings that ends with '11'.

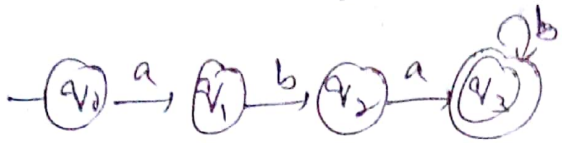
011, 11, 111, 0011 ... ∞'s

Sol:



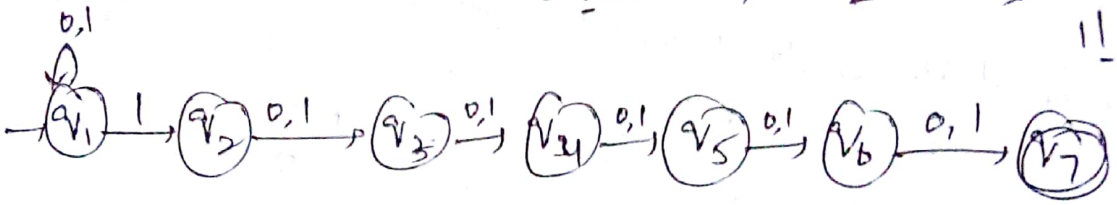
Ex: Design NFA for $\{abab^n / n>=0\}$

Sol:



Ex: Design NFA for the set of strings such that 6th symbol from right end is 1.

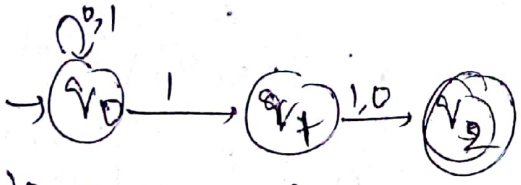
0100000, 0101010, 0001100, 1001100



Ex: Design NFA accepting the set of all strings whose second last symbol is 1.

Sol:

$L = \{110, 010, 111, 0010, 11010 \ldots\}$



$M = \{Q, \Sigma, \delta, q_0, F\}$

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

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

$\delta,$

$q_0 = \{q_0\}$

$F = \{q_2\}$

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

Ex:- Design the NFA transition diagram for the transition table as given below.

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

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

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

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

δ :

$q_0 = \{q_0\}$

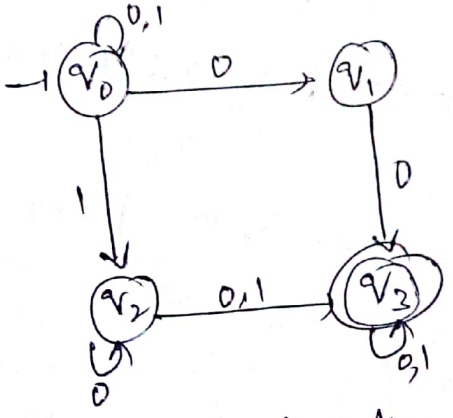
$F = \{q_3\}$.

Sol:- Transition diagram can be drawn by using the mapping function as given in table

$\delta(q_0, 0) = \{q_0, q_1\}$ $\quad \delta(q_1, 0) = \{q_3\}$

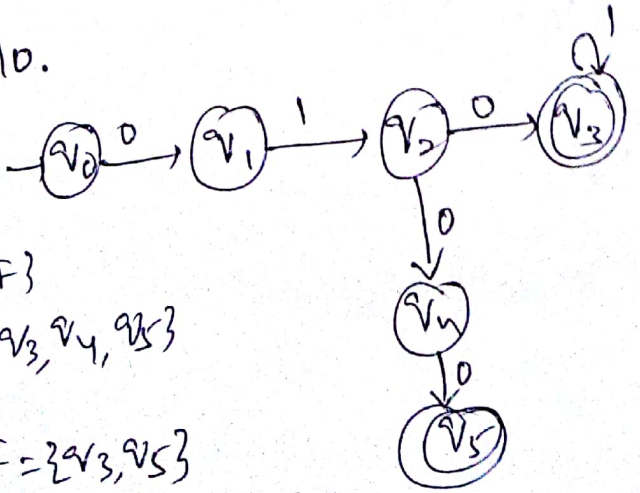
$\delta(q_0, 1) = \{q_0, q_2\}$ $\quad \delta(q_2, 0) = \{q_2, q_3\}$, $\delta(q_2, 1) = \{q_3\}$

$\delta(q_3, 0) = \{q_3\}$ $\quad \delta(q_3, 1) = \{q_3\}$



Ex:- Construct NFA for the language $L = \{0101^n \cup 0100 \mid n \geq 0\}$

Sol:- Here in language L, the first three symbols are common i.e 010.



$M = \{Q, \Sigma, \delta, q_0, F\}$

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

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

$q_0 = \{q_0\}$, $\quad F = \{q_3, q_5\}$

Ex:- Construct a transition diagram for the NDFA (NFA)

$M = (\{q_1, q_2, q_3\}, \{0,1\}, \delta, q_1, \{q_3\})$ where δ is given by :-

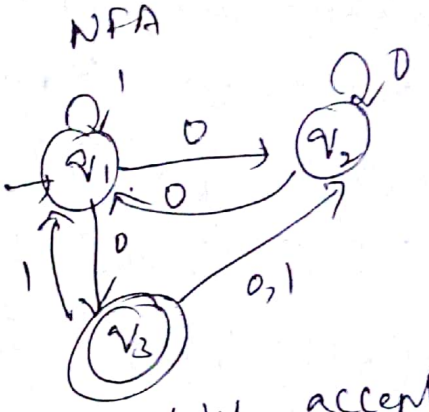
$\delta(q_1, 0) = \{q_2, q_3\}$ $\quad \delta(q_1, 1) = \{q_1\}$

$\delta(q_2, 0) = \{q_1, q_2\}$ $\quad \delta(q_2, 1) = \{\emptyset\}$

$\delta(q_3, 0) = \{q_2\}$ $\quad \delta(q_3, 1) = \{q_1, q_2\}$.

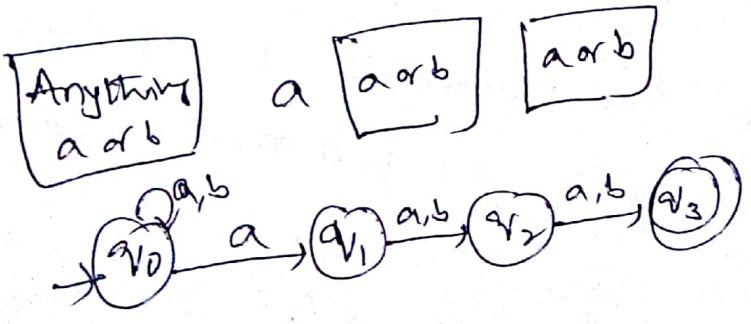
Sol:- Firstly we will design a transition table using the mapping function δ.

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



Ex:- Construct a NFA for a language L which accepts all the strings in which the third symbol from right end is always a. over $\Sigma = \{a, b\}$.

Sol:- The strings in such a language are of the form



NFA

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

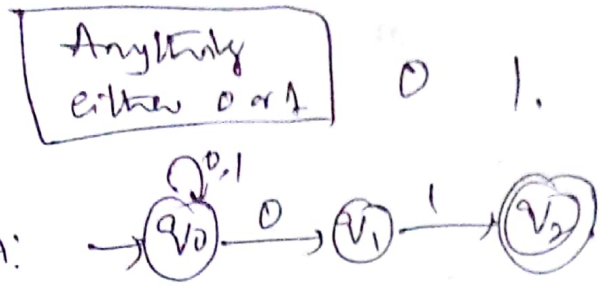
$\Sigma = \{0, 1\}$

$q_0 = \{q_0\}$

$F = \{q_3\}$

Ex: Design NFA accepting all strings ending with 01. over $\Sigma = \{0,1\}$.

Sol:



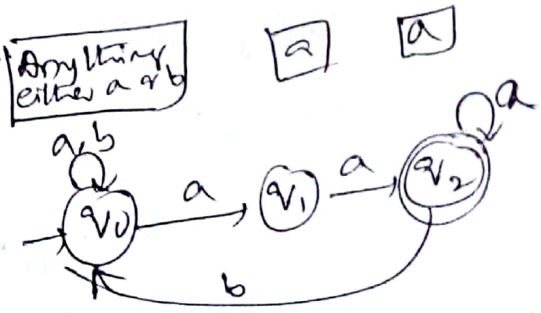
NFA:

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{0,1\}$
$q_1 = \{q_0\}$
$F = \{q_2\}$

Ex: Design NFA to accept strings with a's and b's such that the string end with 'aa'

Sol: The simple FA which accept a string with 'aa' is.

Now there can be a situation where is



$M = (\{q_0, q_1, q_2\}, \{a,b\}, \delta, \{q_0\}, \{q_2\})$

String: aaa

$\delta(q_0, aaa) \vdash \delta(q_0, aa)$
$\vdash \delta(q_0, a)$
$\vdash \delta(q_1, \epsilon)$
not a final state

String aaa

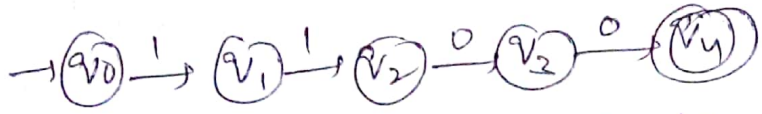
$\delta(q_0, aaa) \vdash \delta(q_0, aa)$
$\vdash \delta(q_1, a)$
$\vdash \delta(q_2, \epsilon)$ ✓ final state

Thus, there are two possibilities, by which we move with string 'aaa' in above given NFA

Ex:- Construct a NFA in which double '1' is followed by double '0'. over $\Sigma = \{0, 1\}$ } 1100, 011001, 011000.

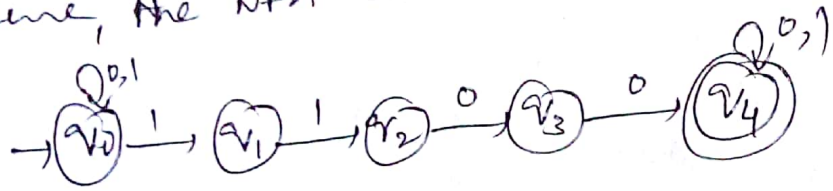
Sol:- The FA with double 1 is drawn below '1'

it should be immediately followed by double '0'. ie 00



Now before double 1 there can be any string of 0 and 1.
Similarly after double 0 there can be any of 0 and 1.

Hence, the NFA becomes



	0	1
→ q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\{q_3\}$	$\emptyset$
q_3	$\{q_4\}$	$\emptyset$
(*) q_4	$\{q_4\}$	$\{q_4\}$

String : 11100

$\delta(q_0, 11100) \vdash \delta(q_0, 1100)$
$\vdash \delta(q_0, 100)$
$\vdash \delta(q_1, 00)$
$\vdash$ Reject

$\delta(q_0, 11100) \vdash \delta(q_0, 1100)$
$\vdash \delta(q_1, 100)$
$\vdash \delta(q_2, 00)$
$\vdash \delta(q_3, 0)$
$\vdash \delta(q_4, \in)$ Accept

Equivalence of DFA's and NFA's

Theorem: Let L be a set accepted by a NFA. Then there exists a DFA that accepts L.

Proof:- Let $M=(Q,\Sigma,\delta,q_0,F)$ be a NFA accepting L.

Define DFA $M'=(Q',\Sigma',\delta',q_0',F')$ as follows

The states of M' are all the subsets of set of states of M, that is, $Q'=2^Q$.

If $Q=\{A,B\}$, then $Q'=\{\emptyset,[A],[B],[AB]\}$.

F' is the set of all states in Q' containing a final state of M. If $F=\{B\}$, then $F'=\{[B],[AB]\}$.

An element Q' will be denoted by $[q_1,q_2,\dots q_i]$ where $q_1,q_2\dots q_i$ are in Q.

NOTE:- $[q_1,q_2,\dots q_i]$ is a single state of DFA corresponding to the set of states of NFA.

$$q_0'=[q_0]$$

we define $\delta'([q_1,q_2,\dots q_i],a)=[p_1,p_2,\dots p_j]$

if $\delta(\{q_1,q_2,\dots q_i\},a)=\{p_1,p_2,\dots p_j\}$.

that is δ' is applied to an element $[q_1,q_2,\dots q_i]$ of Q', it is computed by applying δ to each state of Q represented by $[q_1,q_2,\dots q_i]$.

It is easy to show by induction on length of the input string x that

$$\delta'(q_0', x) = [q_1, q_2, \cdots q_i]$$

iff $\delta(q_0, x) = \{q_1, q_2, \cdots q_i\}$

Basis: The result is trivial for $|x| = 0$, As $\delta(q_0', \epsilon) = [q_0]$

$\Rightarrow q_0' = [q_0]$.

Induction: Suppose that the hypothesis is true for inputs of length m or less. Let 'xa' be string of length m+1 with 'a' in Σ.

Then $\delta'(q_0, xa) = \delta'(\delta'(q_0, x), a)$

By induction hypothesis

$$\delta'(q_0 x) = [p_1, p_2, \cdots p_j]$$

iff $\delta(q_0, x) = \{p_1, p_2, \cdots p_j\}$

By But by the definition of δ'

$$\delta'([p_1, p_2, \cdots p_j], a) = [r_1, r_2, \cdots r_k]$$

Thus $\delta'(q_0', xa) = [r_1, r_2, \cdots r_k]$

iff $\delta(q_0, xa) = \{r_1, r_2, \cdots r_k\}$.

- This establishes the inductive hypothesis.

Now we have to prove that the $L(M) = L(M')$.

- The string x is accepted by NFA or DFA only if it is in one of the final states.

for a string x in NFA, let $\delta(q_0, x) = P$ where $P \in F$, then $\delta'(q_0, x) = [P]$ where $[P] \in F'$, Hence the string x is accepted by it if it is accepted by NFA.

Note :-

* Every language that can be described by NFA can be described by some DFA.
* DFA in practice has more states than NFA.
* DFA equivalent to NFA can have at most 2^n states whereas NFA has only n states.

Conversion of NFA (M_N) to DFA (M_D).

- Let $M_N = (Q_N, \Sigma_N, \delta_N, q_{0N}, F_N)$ be the given NFA to construct equivalent DFA M_D.
- Define M_D as follows

(i) $Q_D = 2^{Q_N}$ If NFA has n states, then DFA can have at most 2^n states.

ii) $\Sigma_D = \Sigma_N$

iii) $[q_{0D}] = \{q_{0N}\}$

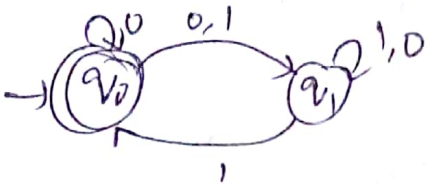
(iv) F_D = set of all the states of Q_D that contains at least one of the final states of F_N.

(v) $\delta_D([q_1, q_2, q_3], a) = \delta_N(q_1, a) \cup \delta_N(q_2, a) \cup \delta_N(q_3, a)$

$= \{p_1, p_2, p_3\}$ say

add the state $[p_1, p_2, p_3]$ to Q_D if it is not there.

Convert the following NFA to DFA



Sol:-

NFA;

$M = \{Q, \Sigma, \delta, q_0, F\}$

	0	1
$\rightarrow^* q_0$	$\{q_0, q_1\}$	$\{q_1\}$
q_1	$\{q_1\}$	$\{q_0, q_1\}$

Step: (1) find the possible set of states Q

Then $2^{\log_2 2^2} = \{4\}$ states, and it is the set of all subsets of q_0, q_1.

$: \{\emptyset, [q_0], [q_1], [q_0 q_1]\}$

Step: (2) find the initial states.

$[q_0] = \{q_0$.

Step: (3) Define the transitions on 0,1 on each state

$\delta[q_0, 0) = [q_0, q_1]$

$\delta(q_0, 1) = [q_1]$

$\delta(q_1, 0) = [q_1]$

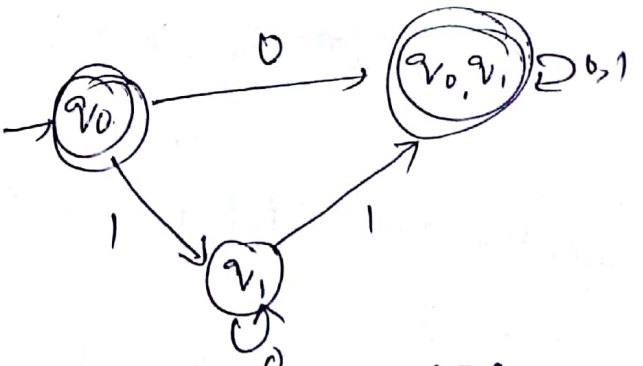
$\delta(q_1, 1) = [q_0, q_1]$

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

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

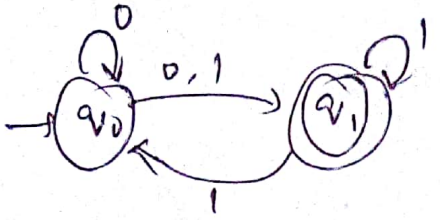
F = The set of states that contain, q_0 called the final states in DFA. i.e, $[q_0]$, $[q_0, q_1]$

	0	1
$\rightarrow^* q_0$	$[q_0, q_1]$	$[q_1]$
q_1	$[q_1]$	$[q_0, q_1]$
$^*[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$



Equivalent DFA.

Ex: Construct DFA, equivalent to the NFA for the below diff.



	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_1\}$
q_1	$\emptyset$	$\{q_0, q_1\}$

Sol:

① No. of states in NFA = 2

No of states in DFA = 4

Set of states are $\{\emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}\}$

Initial state : $\{q_0\}$

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

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

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

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

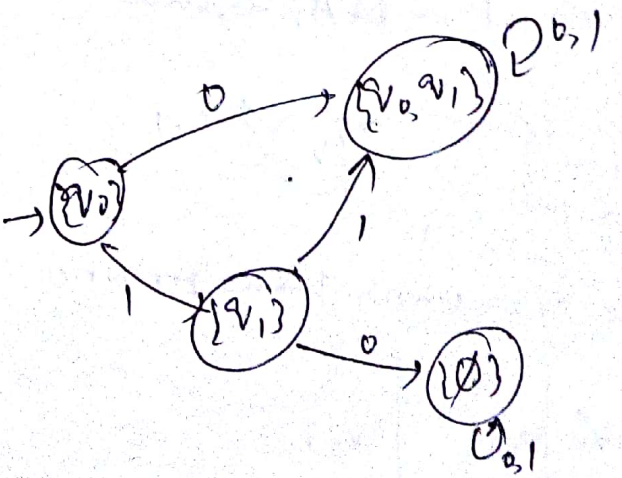
$\delta([q_0, q_1], 0) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \emptyset = \{q_0, q_1\}$

$\delta([q_0, q_1], 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\}$.

F: The set of final states that contain q_1, called final states in DFA. So, $[q_1]$, $[q_0, q_1]$ are the final states in DFA

②

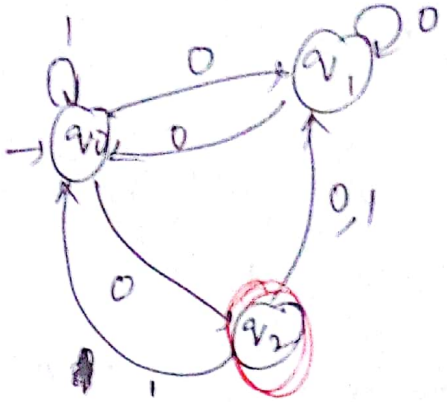
	0	1
→ $[q_0]$	$[q_0, q_1]$	$[q_1]$
* $[q_1]$	$\emptyset$	$[q_0, q_1]$
* $[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$
$\emptyset$	$\emptyset$	$\emptyset$



Convert the following NFA to DFA

NFA

δ	0	1
→ q_0	$\{q_1, q_2\}$	$\{q_0\}$
q_1	$\{q_0, q_1\}$	$\emptyset$
* q_2	$\{q_1\}$	$\{q_0, q_1\}$

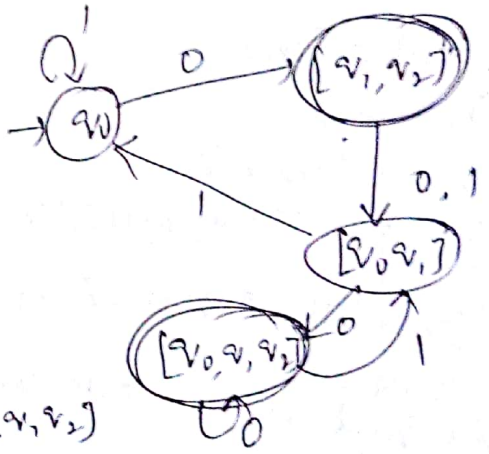


DFA : $Q = 2^3 = 8$

Q has $2^3 = (8)$ states, and it is the set of all subsets of q_0, q_1, q_2

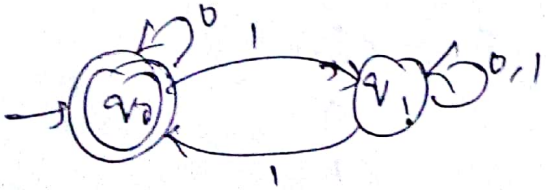
are $\{\emptyset, [q_0], [q_1], [q_2], [q_0, q_1], [q_0, q_2], [q_1, q_2], [q_0, q_1, q_2]\}$

	0	1
→ q_0	$\{q_1, q_2\}$	$\{q_0\}$
* $[q_1, q_2]$	$\{q_0, q_1\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0\}$
* $\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$



final states are $[q_1, q_2]$ & $[q_0, q_1, q_2]$

Ex: Construct a DFA, equivalent to NFA for below fig



Sol: The transition table for NFA

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

	0	1
→ * q_0	$\{q_0\}$	$\{q_1\}$
q_1	$\{q_1\}$	$\{q_0, q_1\}$

$Q = \{q_0, q_1\}$

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

$q_0 = \{q_0\}$

$F = \{q_0\}$

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

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

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

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

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

δ	0	1
$\to \{q_0\}^*$	$\{q_0\}$	$\{q_1\}$
$\{q_1\}$	$\{\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_1\}$

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

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

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

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

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

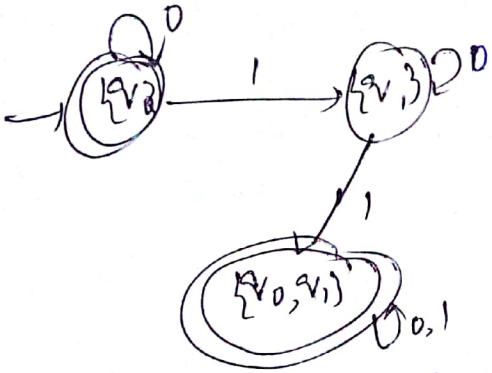
$\delta(\{q_0, q_1\}, 1) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_1\} \cup \{q_0, q_1\}$
$= \{q_0, q_1\}$

~~b) the transition diagram :-~~

The equivalent DFA is $D = (Q', \Sigma', \delta', q_0', F')$,

where $Q' = \{\{q_0\}, \{q_1\}, \{q_0, q_1\}\}$.

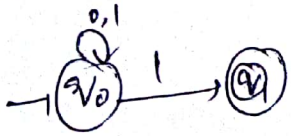
$q_0' = \{q_0\}$, $F' = \{\{q_0\}, \{q_0, q_1\}\}$



equivalent DFA

$L = \{$set of all strings over $(0,1)$ that ends with '1'$\}$

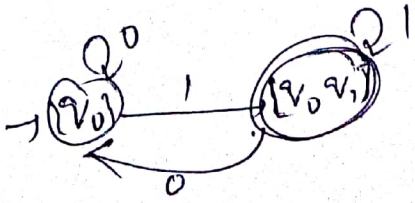
NFA $\Sigma = \{0, 1\}$



	0	1
$\to q_0$	$\{q_0\}$	$\{q_0, q_1\}$
q_1^*	$\emptyset$	$\emptyset$

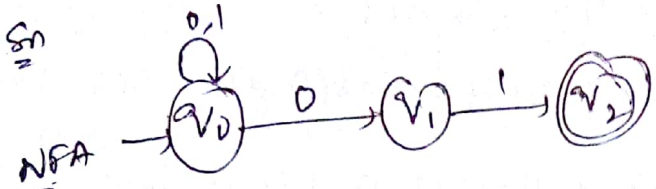
DFA

	0	1
$\to q_0$	$\{q_0\}$	$\{q_0, q_1\}$
$* \{q_0, q_1\}$	$\{q_0\}$	$\{q_0, q_1\}$



Conversion of NFA to DFA

Given below is the NFA for a language, L = { set of all strings over (0,1) that ends with '01' }. Construct its equivalent DFA:



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

DFA

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

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

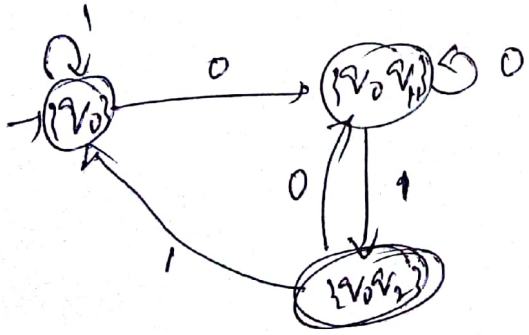
$\delta(q_0, 1) = \{q_0\}$

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

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

$\delta(\{q_0, q_2\}, 0) = \delta(q_0, 0) \cup \delta(q_2, 0)$
$= \{q_0, q_1\} \cup \{\emptyset\}$
$= \{q_0, q_1\}$

$\delta(\{q_0, q_2\}, 1) = \delta(q_0, 1) \cup \delta(q_2, 1)$
$= \{q_0\} \cup \{\emptyset\} = \{q_0\}$



Finite Automata with ∈-Transitions

T-F

NFA with ∈-moves:

We can extend a NFA by introducing '∈-moves' that allow us to make a transition on the empty string. There would be an edge labelled ∈ between two states and this edge allows transitions from one state to another without receiving an input symbol.

" A finite automata that is modified to permit transitions without input symbols, and with zero, one, or more transitions on input symbols, is an NFA with ∈-moves. (transitions).

NFA with ∈-moves transitions is a five-tuple:

$$NFA\text{-}\epsilon = (Q, \Sigma, \delta, q_0, F)$$

Where

- Q is a finite set of input states
- Σ is a set of input symbols
- δ is a transition function such that
$$\delta : Q \times (\Sigma \cup \epsilon) \to 2^Q$$
- q_0 is initial state $q_0 \in Q$
- F is final state. $F \subseteq Q$.

Ex:- Below fig gives NFA with ∈ transitions and it accepts strings of the form $\{0^m 1^n 2^o \mid m,n,o \geq 0\}$, that is any number of 0's followed by any number of 1's followed by any number of 2's.

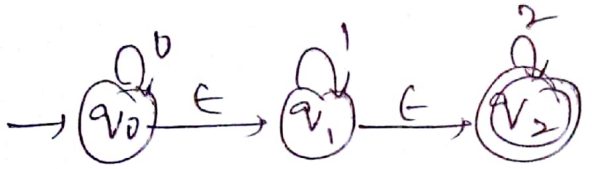


fig: NFA with ∈-transitions.

	0	1	2	∈
→ q_0	q_0	∅	∅	q_1
q_1	∅	q_1	∅	q_2
*q_2	∅	∅	q_2	∅

Epsilon closure (∈-closure)

Epsilon closure of a state is simply the set of all states that we can reach by ∈ input. This is denoted by either

$\hat{\epsilon}(q)$ or ∈-closure(q).

Eg: For above example, epsilon closure of q_0, q_1, q_2 states are

∈-closure(q_0) = $\{q_0, q_1, q_2\}$ ⇒ self state + ∈-reachable state

∈-closure(q_1) = $\{q_1, q_2\}$

∈-closure(q_2) = $\{q_2\}$

Q: Design an NFA with ϵ-moves to accept all the strings with any number of a's followed by any number of b's followed by any number of c's.

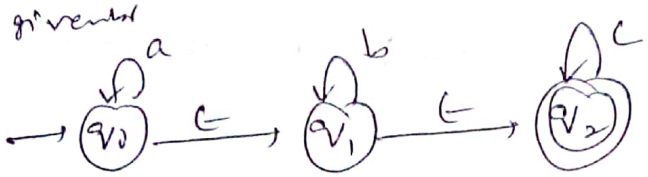
Sol: We are to design an NFA-ϵ for the regular expression $r = a^*b^*c^*$, i.e. NFA-ϵ, M is to be designed to accept the language of r.

i.e. $L(M) = \{\epsilon, ab, bb, cc, abc, aabbcc, \dots\}$

Consider $\Sigma = \{a, b, c\}$, $Q = \{q_0, q_1, q_2\}$

q_0 = initial state, $F = q_2$.

δ is given by



Transition table:

Q \ Σ	a	b	c	ϵ
→ q_0	$\{q_0\}$	$\emptyset$	$\emptyset$	$\{q_1\}$
q_1	$\emptyset$	$\{q_1\}$	$\emptyset$	$\{q_2\}$
*q_2	$\emptyset$	$\emptyset$	$\{q_2\}$	$\emptyset$

NFA-ϵ acting for the input string

To show that string abc is accepted by NFA-ϵ

→ To compute ϵ-closure

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

ϵ-closure$(\{q_1\}) = \{q_1, q_2\}$

ϵ-closure$(\{q_2\}) = \{q_2\}$

To compute δ: abc

$\delta(\epsilon\text{-closure}(\{q_0\}), a) = \delta(\{q_0, q_1, q_2\}, a)$

$= \delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a)$

$= \{q_0\} \cup \{\emptyset\} \cup \{\emptyset\} = \{q_0\}$

$\delta(\epsilon\text{-closure}(\{q_0\}), ab) = \delta(\epsilon\text{-closure}(\delta(\epsilon\text{-closure}(\{q_0\}), a)), b)$

$= \delta(\epsilon\text{-closure}(\{q_0\}), b)$

$= \delta(\{q_0, q_1, q_2\}, b)$

$= \delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b)$

$= \{\emptyset\} \cup \{q_1\} \cup \{\emptyset\} = \{q_1\}$

$\delta(\epsilon\text{-closure}(\{q_0\}), abc) = \delta(\epsilon\text{-closure}(\delta(\epsilon\text{-closure}(\{q_0$

$= \delta(\epsilon\text{-closure}(\{q_1, 3\}), c)$

$= \delta(\{q_1, q_2, 3\}, c)$

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

$= \{\emptyset\} \cup \{q_2\} = \underline{\{q_2\}}$

- $\epsilon\text{-closure}(\delta(\epsilon\text{-closure}(\{q_0\}), abc)) = \epsilon\text{-closure}(\{q_2\})$

$= \underline{\{q_2\}}$

Since $\{q_2\}$ is accepting state, hence abc is accepted.

* $\hat{\delta}(q, \epsilon) = \epsilon\text{-closure}(q)$

$\hat{\delta}(q, x) = \epsilon\text{-closure}(\delta(\hat{\delta}(q, \epsilon), x))$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q), x))$

Conversion of 'NFA with ϵ-Transitions' to NFA without ϵ-transitions.

Let $N = (Q, \Sigma, \delta, q_0, F)$ is a NFA with ϵ-transition then there exist $N' = (Q, \Sigma, \hat{\delta}, q_0, F')$ is a NFA without ϵ-Transitions.

Conversion Algorithm :-

(i) First find the ϵ-closure of all the states in the device.

ii) Calculate extended transition function, using following conversion formula

~~$\hat{\delta}(q, x\epsilon) = \epsilon\text{-closure}(\delta(\hat{\delta}(q, \epsilon), x))$~~

i) $\hat{\delta}(q, \epsilon) = \epsilon\text{-closure}(q)$

ii) $\hat{\delta}(q, x) = \epsilon\text{-closure}(\delta(\hat{\delta}(q, \epsilon), x))$

iii) 'F' is a set of all states, whose ϵ-closure containing a finite state in F.

$F' = \begin{cases} F \cup \{q_0\} & \text{if } \epsilon\text{-closure}(q_0) \text{ contain a member of } F \\ F' = F & \text{otherwise.} \end{cases}$

Find an equivalent NFA without ε-transitions for NFA with ε-transitions shown in below figure.

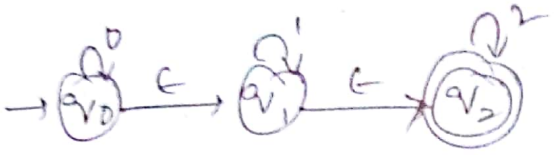


fig: NFA with ε transitions.

Sol:-

Transition table

	0	1	2	ε
→ q_0	q_0	∅	∅	q_1
q_1	∅	q_1	∅	q_2
* q_2	∅	∅	q_2	∅

Step 1: find ε-closure for each state

$$\epsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}$$
$$\epsilon\text{-closure}(q_1) = \{q_1, q_2\}$$
$$\epsilon\text{-closure}(q_2) = \{q_2\}$$

Step 2: find the transitions on each state for each element.

$$\hat{\delta}(q_0, 0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), 0))$$
$$= \epsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 0))$$
$$= \epsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0))$$
$$= \epsilon\text{-closure}(\{q_0\} \cup \{\emptyset\} \cup \{\emptyset\})$$
$$= \epsilon\text{-closure}(\{q_0\})$$
$$= \{q_0, q_1, q_2\}.$$

$$\hat{\delta}(q_0, 1) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), 1))$$
$$= \epsilon\text{-closure}(\delta(\{q_0, q_1, q_2\}, 1))$$
$$= \epsilon\text{-closure}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1))$$
$$= \epsilon\text{-closure}(\{\emptyset\} \cup \{q_1\} \cup \{\emptyset\})$$
$$= \epsilon\text{-closure}(\{q_1\})$$
$$= \{q_1, q_2\}.$$

$\hat{\delta}(q_0, 2) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), 2))$

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

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

$= \epsilon\text{-closure}(\emptyset \cup \{\emptyset\} \cup \{q_2\})$

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

$= \{q_2\}$

$\hat{\delta}(q_1, 0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1), 0))$

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

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

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

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

$= \{\emptyset\}$

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

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

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

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

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

$= \{q_1, q_2\}$

$\hat{\delta}(q_1, 2) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1), 2))$

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

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

$= \epsilon\text{-closure}(\{\emptyset\} \cup \{q_2\})$

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

$= \{q_2\}$

$=$

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

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

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

$= \{\emptyset\}$

$\hat{\delta}(q_2,1) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_2),1))$

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

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

$= \{\emptyset\}$

$\hat{\delta}(q_2,2) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_2),2))$

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

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

$= \{q_2\}$

$\epsilon\text{-closure}(q_0) = \{q_0,q_1,q_2\}$

$\epsilon\text{-closure}(q_1) = \{q_1,q_2\}$

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

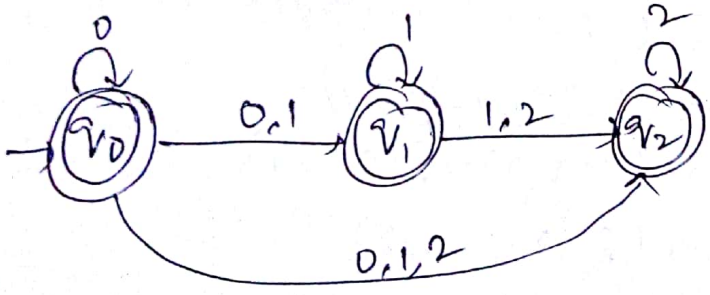
NFA without ϵ-transitions is

Step 3: Identify the final states

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

Here q_0, q_1, and q_2 is a final state because $\epsilon\text{-closure}(q_0)$, $\epsilon\text{-closure}(q_1)$ and $\epsilon\text{-closure}(q_2)$ contains final state q_2.

	0	1	2
→ *q_0	$\{q_0,q_1,q_2\}$	$\{q_1,q_2\}$	$\{q_2\}$
*q_1	$\emptyset$	$\{q_1,q_2\}$	$\{q_2\}$
*q_2	$\emptyset$	$\emptyset$	$\{q_2\}$



Transition diagram for NFA without ϵ-transition

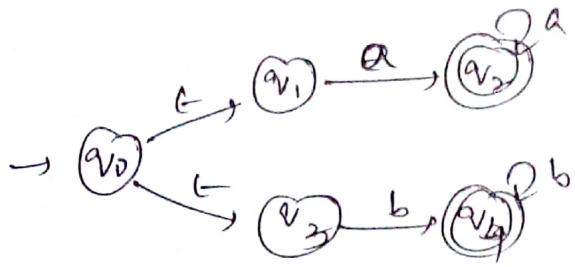
Design a NFA with ϵ-moves to accept the set of all strings over $\Sigma = \{a, b\}$ that ends with a or b, followed by any number of a's or for any number of b's.

Sol: We are to design NFA with ϵ for R.E (r) = $aa^* + bb^*$.

$L(M) = \{a, b, aa, bb, aaa, bbb, \ldots\}$

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

$L(M) = \{w \in \Sigma^* \mid$ w is either a or any no. of a's or b or any number of bs $\}$



	a	b	ϵ
→ q_0	$\emptyset$	$\emptyset$	$\{q_1, q_3\}$
q_1	q_2	$\emptyset$	$\emptyset$
q_2	q_2	$\emptyset$	$\emptyset$
q_3	$\emptyset$	q_4	$\emptyset$
q_4	$\emptyset$	q_4	$\emptyset$

Transition Table

(1) Step 1.

ϵ-closure(q_0) = $\{q_0, q_1, q_3\}$

ϵ-closure(q_1) = $\{q_1\}$

ϵ-closure(q_2) = $\{q_2\}$

ϵ-closure(q_3) = $\{q_3\}$

ϵ-closure(q_4) = $\{q_4\}$

String aa

$\delta(\epsilon\text{-closure}(\{q_0\}), a) = \delta(\{q_0, q_1, q_3\}, a)$

$= \delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_3, a)$

$= \{\emptyset\} \cup \{q_2\} \cup \{\emptyset\}$

$= \{q_2\}$

$\delta(\epsilon\text{-closure}(\{q_0\}), aa) = \delta(\epsilon\text{-closure}(\delta(\epsilon\text{-closure}(\{q_0\}), a)), a)$

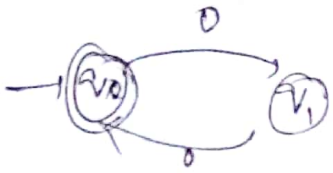
$= \delta(\epsilon\text{-closure}(\{q_2\}), a)$

$= \delta(q_2, a)$

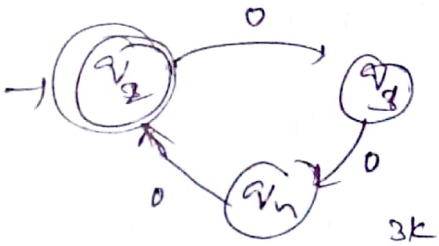
$= \{q_2\}$ final state

Ex:- Design NFA with ε for language $L = \{0^k \mid k \text{ is multiple of } 2 \text{ or } 3\}$

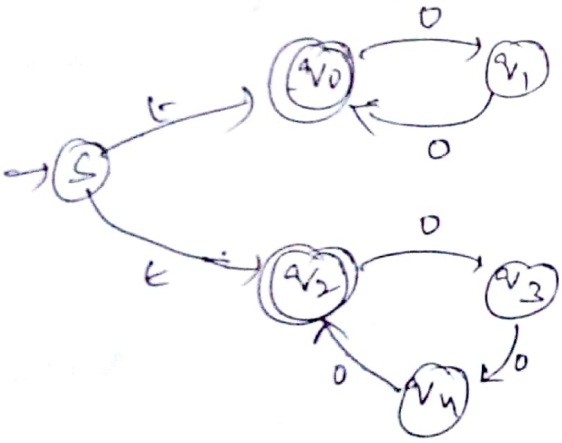
Sol:



NFA for 0^{2k}



NFA for 0^{3k}



ε- closure (S) = {S, q_0, q_2}

ε- closure (q_1) =

Minimization of DFA

I - G

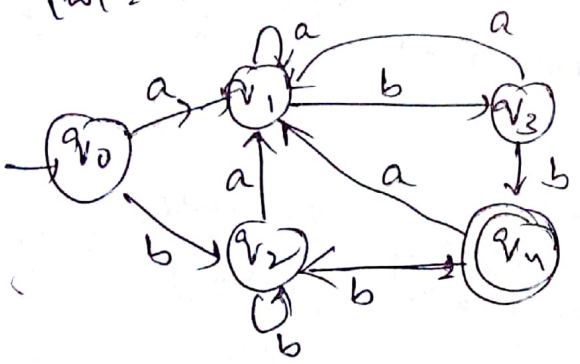
two states (P, q) are equivalent if

i) $\delta(P, w) \in F \Rightarrow \delta(q, w) \in F$ ✓

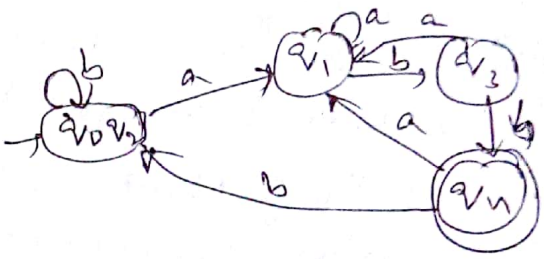
ii) $\delta(P, w) \notin F \Rightarrow \delta(q, w) \notin F$

$|w| = 0$, 0 equivalent
$|w| = 1$, 1 equivalent
$|w| = 2$, 2 equivalent
$|w| \geq n$: n equivalent

Ex:



minimized DFA



Step ① Try to delete all the states which are not reachable from the initial state.

Step ② Draw the state transition table.

	a	b
→ q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	*q_4
*q_4	q_1	q_2

① ○ - equivalent sets

[non-final states] [final states]

$[q_0, q_1, q_2, q_3]$ $\{q_4\}$

② 1 - equivalent $(q_0, q_1) =$ $(q_0, q_2) =$ $(q_2, q_3) = q_2 \neq q_3$

$[q_0, q_1, q_2]$ $[q_3]$ $[q_4]$

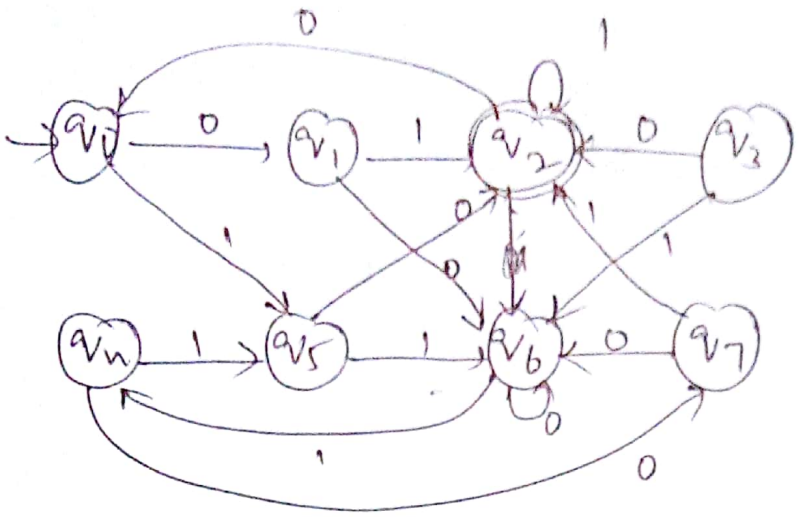
③ 2 - equivalent $(q_0, q_1) = q_0 \neq q_2$ $(q_0, q_2) =$

$[q_0, q_2]$, $[q_1]$, $[q_3]$ $[q_4]$

④ 3 - equivalent (q_0, q_2)

$[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

Ex: Minimize DFA



	0	1
→ q_0	q_1	q_5
q_1	q_6	q_2
* q_2	q_0	q_2
q_3	q_2	q_6
q_4	q_1	q_5
q_5	q_2	q_6
q_6	q_6	q_4
q_7	q_6	q_2

State ① Remove states which are not reachable from initial state

- So $[q_3]$ is not reachable from initial state, so ~~remove all states~~

State ② 0 equivalent - non final state | final state

$[q_0, q_1, q_2, q_4, q_5, q_6, q_7]$ $[q_2]$

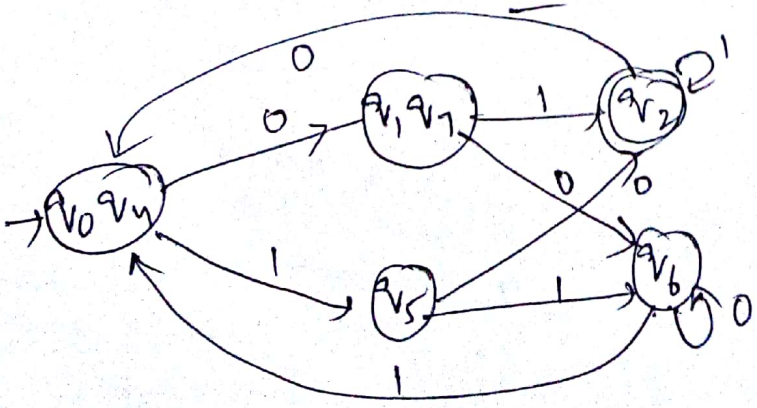
1 equivalent - ~~$[q_0, q_1, q_4, q_6]$ $[q_5, q_7]$ $[q_2]$~~

$[q_0, q_4, q_6]$ $[q_1, q_7]$ $[q_5]$ $[q_2]$

2 equivalent - $[q_0, q_4]$ $[q_6]$ $[q_1, q_7]$ $[q_5]$ $[q_2]$

3 equivalent - $[q_0, q_4]$ $[q_6]$ $[q_1, q_7]$ $[q_5]$ $[q_2]$

4 equivalent - $[q_0, q_4]$ $[q_6]$ $[q_1, q_7]$ $[q_5]$ $[q_2]$



Minimization of Finite Automata:

- for any deterministic automaton, that has more number of states, we can construct equivalent DFA with less number of states.

Construction of Minimum Automaton:

(a) **Step 1**: Identify the unreachable states and from input state, eliminate them from DFA

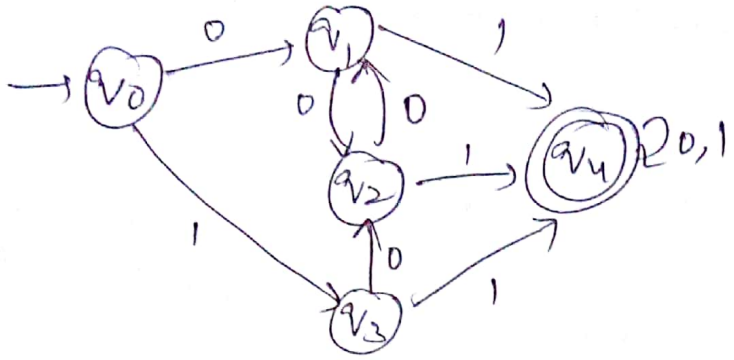
∴ **Step 2**: (Construction of Π_0) By definition of 0-equivalence, $\Pi_0 = \{Q_1^0, Q_2^0\}$ where Q_1^0 is the set of all final states and $Q_2^0 = Q - Q_1^0$.

⇒ **Step 3**: (Construction of Π_{k+1} from Π_k). Let Q_i^k be any subset in Π_k. If q_1 and q_2 are in Q_i^k, They are $(k+1)$-equivalent provided $\delta(q_1, a)$ and $\delta(q_2, a)$ are k-equivalent. Find out whether $\delta(q_1, a)$ and $\delta(q_2, a)$ are in the same equivalence class in Π_k for $a \in \Sigma$. If so, q_1 and q_2 are $(k+1)$-equivalent. In this way, Q_i^k is further divided into $(k+1)$-equivalence classes. Repeat this for every Q_i^k in Π_k to get all the elements of Π_{k+1}.

Step 4: Construct Π_n for $n = 1, 2, \ldots$ until $\Pi_n = \Pi_{n+1}$

Step 5: (Construction of minimum automaton). For the required minimum state automaton, the states are the equivalence classes obtained in Step 4. i.e. the elements of Π_n. The state table is obtained by replacing a state q by the corresponding equivalence class $[q]$.

Ex:- Construct a minimum state automaton equivalent to the following finite automata:



Sol:- Transition table for given DFA

	0	1
→ q_0	q_1	q_3
q_1	q_2	q_4
q_2	q_1	q_4
q_3	q_2	q_4
q_4	q_4	q_4

Construction of minimum Automata

Step 1: identify the unreachable states from initial state and eliminate them from DFA.

- Here, all the states are reachable from input state q_0. So, no need to eliminate any state from the given DFA.

Step 2: Construction of 'Π_0'.

Let DFA $M = \{\{q_0, q_1, q_2, q_3, q_4\}, \{0,1\}, \delta, q_0, \{q_4\}\}$

Set of final states $Q_1^0 = \{q_4\}$

Set of non-final states, $Q_2^0 = \{q_0, q_1, q_2, q_3\}$

So, we know that $\Pi_0 = \{Q_1^0, Q_2^0\}$

i.e $\Pi_0 = \{\{q_4\}, \{q_0, q_1, q_2, q_3\}\}$

Π_0 - is 0-equivalent class.

Step 3: Construction of Π_1

Consider $Q_1^0 = \{q_4\}$.

We can't partition Q_1^0.

So: $Q_1^0 = \{q_4\}$.

→ Consider $Q_2^1 = \{q_0, q_1, q_2, q_3\}$

w.r $q_0, q_1 \in Q_2^0$

$\delta(q_0, 0) = \{q_1\}$ $\quad \delta(q_0, 1) = \{q_3\}$

$\delta(q_1, 0) = \{q_2\}$ $\quad \delta(q_1, 1) = \{q_4\}$.

The pair of states $\{q_1, q_2\}$ are in ~~same state~~ same subset, and (q_3, q_4) are in different states.

So, we can say that q_0, q_1 are not 1-equivalent..

w.r $q_1, q_2 \in Q_2^1$

$\delta(q_1, 0) = \{q_2\}$ $\quad \delta(q_1, 1) = \{q_4\}$

$\delta(q_2, 0) = \{q_1\}$ $\quad \delta(q_2, 1) = \{q_4\}$

The pair of states under '0' and '1' are

(q_1, q_2) → belongs to same subset.

(q_4, q_4) → belongs to same subset.

↳ q_1, q_2 are 1-equivalent.

Consider $q_2, q_3 \in Q_2$

$\delta(q_2, 0) \to \{q_1\}$ $\quad$ $\delta(q_2, 1) = \{q_4\}$

$\delta(q_3, 0) \Rightarrow \{q_2\}$ $\quad$ $\delta(q_3, 1) = \{q_4\}$

The pair of states under '0' and 1 are

(q_1, q_2) are belongs to same subset

(q_4, q_4) are belongs to same subset

$\therefore$ q_2, q_3 are 1-equivalent.

$\therefore \Pi_1 = \{\{q_4\}, \{q_0\}, \{q_1, q_2, q_3\}\}$ 1-equivalent.

$Q_1' = \{q_4\}$, $Q_2' = \{q_0\}$, $Q_3' = \{q_1, q_2, q_3\}$.

Construction of Π_2

Consider $\Pi_1 = \{\{q_4\}, \{q_0\}, \{q_1, q_2, q_3\}\}$

Let $Q_1' = \{q_4\}$, we can't partition Q_1'.

Let $Q_2' = \{q_0\}$, we can't partition Q_2'.

Let $Q_3' = \{q_1, q_2, q_3\}$

Consider $(q_1, q_2) \in Q_3'$.

$\delta(q_1, 0) = \{q_2\}$ $\quad$ $\delta(q_1, 1) = \{q_4\}$

$\delta(q_2, 0) = \{q_7\}$ $\quad$ $\delta(q_2, 1) = \{q_4\}$.

The pair of states under '0' and '1' are

(q_2, q_7) are belongs to same subset

(q_4, q_4) are belongs to same subset

So, (q_1, q_2) are 2-equivalent

Consider $(q_2, q_3) \in Q_3'$

$\delta(q_2, 0) = \{q_1\}$ $\quad$ $\delta(q_2, 1) = \{q_4\}$

$\delta(q_3, 0) = \{q_2\}$ $\quad$ $\delta(q_3, 1) = \{q_4\}$

The pair of states under '0' and '1' are

(q_1, q_2) are belongs to same subset

(q_4, q_4) are belongs to same subset.

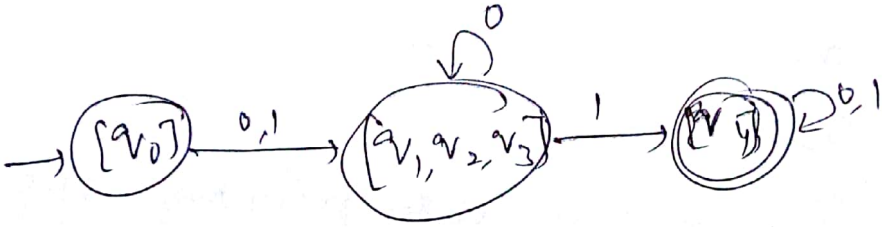
So; (q_2, q_3) are 1- equivalent.

$\Pi_2 = \{\{q_4\}, \{q_0\}, \{q_1, q_2, q_3\}\}$.

If we observe, $\Pi_1 = \Pi_2$, so we stop the process.

∴ The states of required optimised DFA are equivalence classes of 'Π_2'.

Therefore, the optimised DFA is shown in below figure.



∴ So

$M' = \{Q', \Sigma, \delta, q_0', F'\}$

$Q' = \{\{q_0\}, \{q_1, q_2, q_3\}, \{q_4\}\}$

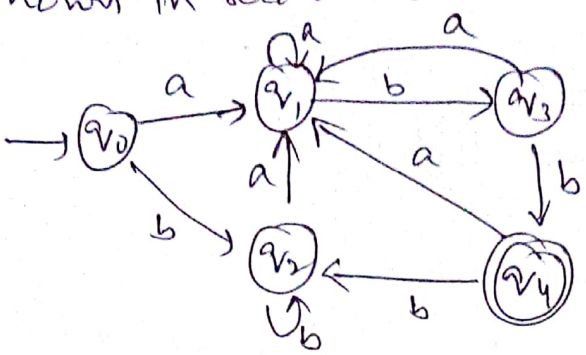
$\Sigma = \{0, 1\}$

$q_0' = \{q_0\}$

$F' = \{q_4\}$

Σ

Ex: Find minimum finite-state automata for the DFA shown in below fig



Sol: Transition table for given DFA

	a	b
q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
q_4	q_1	q_2

Step 1:- Identify the unreachable states from initial state and eliminate them from DFA.

Here, all the states are reachable from initial state q_0, so, no need to eliminate any state from the given DFA.

Step 2: Construction of Π_0 i.e. 0-equivalence sets

Let DFA $M = \{\{q_0, q_1, q_2, q_3, q_4\}, \{a,b\}, \delta, \{q_0\}, \{q_4\}\}$

set of final states $Q_1^0 = \{q_4\}$

set of non-final states $Q_2^0 = \{q_0, q_1, q_2, q_3\}$

So, we know that $\Pi_0 = \{Q_1^0, Q_2^0\}$.

i.e. $\Pi_0 = \{\{q_4\}, \{q_0, q_1, q_2, q_3\}\}$

Π_0 - is '0'-equivalence set

Step 3:

Step 3: Construction of Π_1 i.e '1-equivalence set'

Consider $Q_1^0 = \{q_4\}$, we can't partition Q_1^0.

$Q_1' = \{q_4\}$.

Consider $Q_2^0 = \{q_0, q_1, q_2, q_3\}$

Let $q_0, q_1 \in Q_2^0$

$\delta(q_0, a) = \{q_1\}$ $\quad \delta(q_0, b) = \{q_2\}$

$\delta(q_1, a) = \{q_1\}$ $\quad \delta(q_1, b) = \{q_3\}$.

The pair of states under 'a' and 'b' are

(q_1, q_1) are belongs to same subset

(q_2, q_3) are belongs to same subset.

So, (q_0, q_2) are 1-equivalent.

Let $q_0, q_2 \in Q_2^0$

$\delta(q_0, a) = \{q_1\}$ $\quad \delta(q_0, b) = \{q_2\}$

$\delta(q_2, a) = \{q_1\}$ $\quad \delta(q_2, b) = \{q_2\}$

The pair of states under 'a' and 'b' are

(q_1, q_1) are belongs to same subset

(q_2, q_2) are belongs to same subset.

So, (q_0, q_2) are 1-equivalent.

Let $q_0, q_3 \in Q_2^0$

$\delta(q_0, a) = \{q_1\}$ $\quad \delta(q_0, b) = \{q_2\}$

$\delta(q_3, a) = \{q_1\}$ $\quad \delta(q_3, b) = \{q_4\}$.

The pair of states under 'a' and 'b' are

(q_1, q_1) are belongs to same subset

(q_2, q_4) are not belongs to same subset, so

(q_0, q_3) are not belongs to 1- equivalent set.

Steps:

So $\Pi_1 = \{\{q_4\}, \{q_0, q_1, q_2\}, \{q_3\}\}$,

$Q_1' = \{q_4\}$, $Q_2' = \{q_0, q_1, q_2\}$, $Q_3' = \{q_3\}$.

Step 2: Construction of Π_2

Consider $\Pi_1 = \{\{q_4\}, \{q_0, q_1, q_2\}, \{q_3\}\}$

Let $Q_1' = \{q_4\}$, we can't partition Q_1'

$Q_2' = \{q_3\}$, we can't partition Q_2'

Let $Q_3' = \{q_0, q_1, q_2\}$.

Consider $(q_0, q_1) \in Q_3'$.

$\delta(q_0, a) = \{q_1\}$ $\delta(q_0, b) = \{q_2\}$

$\delta(q_1, a) = \{q_1\}$ $\delta(q_1, b) = \{q_3\}$...

The pair of states under 'a' and 'b' are

(q_1, q_1) belongs to same set

(q_2, q_3) are not belongs to same set so, $q_0 \neq q_1$

so, (q_0, q_1) are not 2-equivalent

Consider $(q_0, q_2) \in Q_3'$

$\delta(q_0, a) = \{q_1\}$ $\delta(q_0, b) = \{q_2\}$

$\delta(q_2, a) = \{q_1\}$ $\delta(q_2, b) = \{q_2\}$

So The pair of states under 'a' and 'b' are

(q_1, q_1) are belongs to same set

(q_2, q_2) are belongs to same set, so,

(q_0, q_2) are belongs to same set $q_0 = q_2$

so if $\{q_0, q_2\}$ are 2-equivalent

So $\Pi_2 = \{\{q_4\}, \{q_3\}, \{q_1\}, \{q_0, q_2\}\}$.

$Q_1^2 = \{q_4\}$, $Q_2^2 = \{q_3\}$ $Q_3^2 = \{q_1\}$, $Q_4^2 = \{q_0, q_2\}$

Step 5: Construction of Π_3

Consider $\Pi_2 = \{\{q_4\}, \{q_2\}, \{q_1\}, \{q_0, q_2\}\}$.

Let $Q_1^2 = \{q_4\}$, we can't partition Q_1^2

$Q_2^2 = \{q_3\}$, we can't partition Q_2^2

$Q_3^2 = \{q_1\}$, we can't partition Q_3^2

let $Q_4^2 = \{q_0, q_2\}$

Consider (q_0, q_2) of Q_4^2

$\delta(q_0, a) = \{q_1\}$ $\quad$ $\delta(q_0, b) = \{q_2\}$

$\delta(q_2, a) = \{q_1\}$ $\quad$ $\delta(q_2, b) = \{q_2\}$

So, The pair of states under 'a' and 'b' are

(q_1, q_1) are belongs to same set,

(q_2, q_2) are belongs to same set. $\Rightarrow q_0 = q_2$

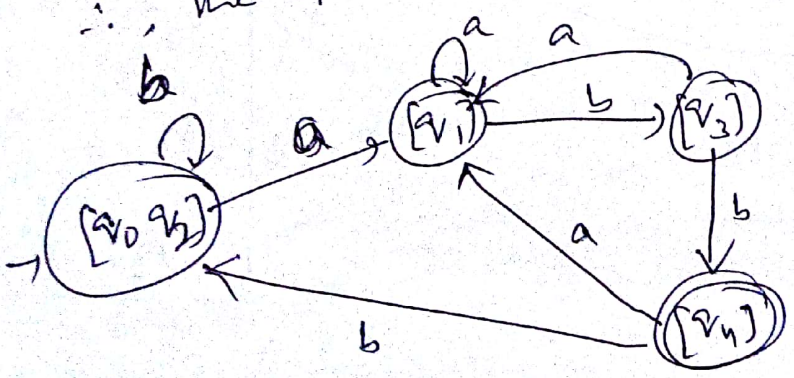
ie $\{q_0, q_2\}$ are 3-equivalent.

$\Pi_3 = \{\{q_4\}, \{q_3\}, \{q_1\}, \{q_0, q_2\}\}$.

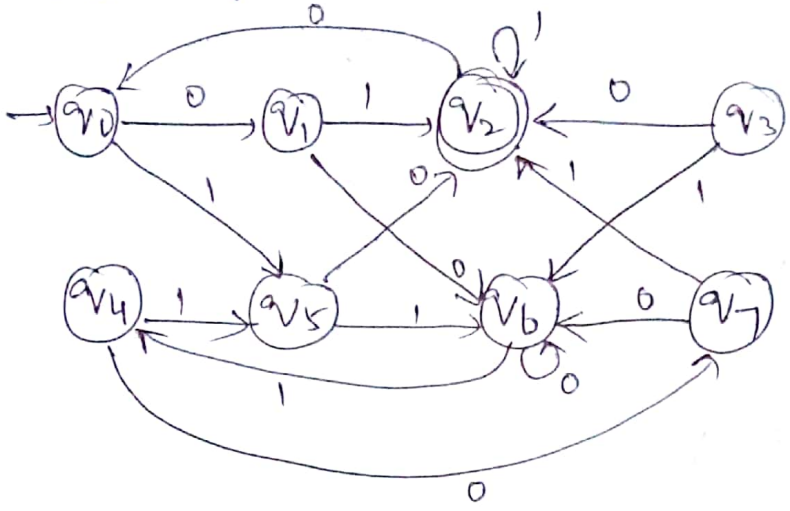
If we observe, $\Pi_2 = \Pi_3$. we stop the process.

The states of required optimized DFA are equivalence classes of Π_3.

∴ The optimized DFA is shown in below figure.



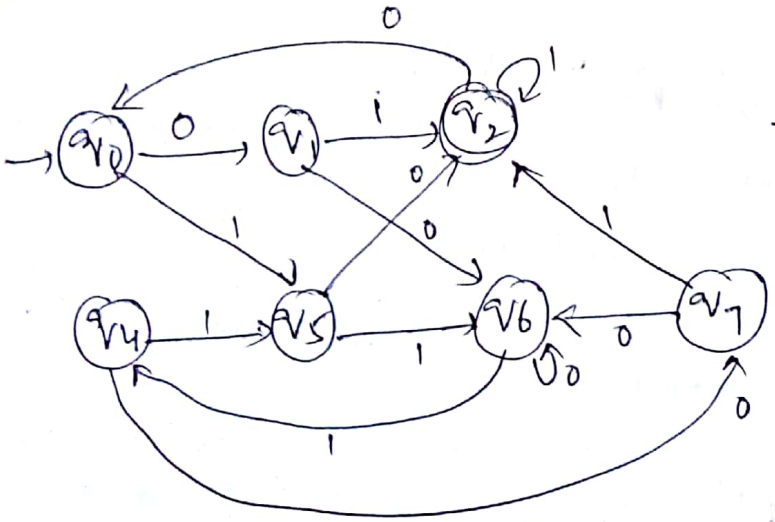
Ex:- Find the minimum finite state automata for the DFA shown in below fig.



Sol:-

Step 1:- Identify the unreachable states from initial state and eliminate them from DFA.

Here, q_3 is the state, it is not reachable from the initial state q_0, so eliminate state q_3 and draw the new DFA without q_3 state.



Transition Table

	0	1
→ q_0	q_1	q_5
q_1	q_6	q_2^*
*q_2	q_0	q_2^*
q_4	q_7	q_5
q_5	q_2^*	q_6
q_6	q_6	q_4
q_7	q_6	q_2^*

Step 2: Construction of Π_0 . ie 0-equivalence sets.

Let DFA $M = \{\{q_0, q_1, q_2, q_4, q_5, q_6, q_7\}, \{0,1\}, \delta, \{q_0\}, \{q_2\}\}$

Set of final states $Q_1^0 = \{q_2\}$

Set of non-final states $Q_2^0 = \{q_0, q_1, q_4, q_5, q_6, q_7\}$

So, we know that $\Pi_0 = \{Q_1^0, Q_2^0\}$

ie $\Pi_0 = \{\{q_2\}, \{q_0, q_1, q_4, q_5, q_6, q_7\}\}$

Π_0 is '0-equivalent' set.

Step 3: Construction of Π_1 , ie 1-equivalence sets

Consider $Q_1^0 = \{q_2\}$ - we can't partition Q_1^0.

Consider $Q_2^0 = \{q_0, q_1, q_4, q_5, q_6, q_7\}$,

Let $q_0, q_1 \in Q_2^0$

$\delta(q_0, 0) = \{q_1\}$ $\delta(q_0, 1) = \{q_5\}$

$\delta(q_1, 0) = \{q_6\}$ $\delta(q_1, 1) = \{q_2\}$.

The pair of states under '0' and '1' are (q_1, q_6) are belongs to same subset, but (q_5, q_2) are belongs to different sets, so, $q_0 \neq q_1$, so they are not 1-equivalent sets

Let $q_0, q_4 \in Q_2^0$

$\delta(q_0, 0) = \{q_1\}$ $\delta(q_0, 1) = \{q_5\}$

$\delta(q_4, 0) = \{q_7\}$ $\delta(q_4, 1) = \{q_5\}$

The pair of states under '0' and '1' are (q_1, q_7) are belongs to same subsets and (q_5, q_5) are also belongs to same subsets) so, q_0, q_4 are 1-equivalent.

let $\{q_0, q_5\} \in Q_2^0$

$\delta(q_0, 0) = \{q_1\}$ $\quad \delta(q_0, 1) = \{q_5\}$

$\delta(q_5, 0) = \{q_2\}$ $\quad \delta(q_5, 1) = \{q_6\}$.

The pair of states under '0' and '1' are

(q_1, q_2) are not belongs to one subset they are belongs to different set.

(q_5, q_6) are belongs to one subset. so, $q_0 \neq q_5$

q_0 and q_5 are not belongs to 1-equivalent class

let $\{q_0, q_6\} \in Q_2^0$

$\delta(q_0, 0) = \{q_1\}$ $\quad \delta(q_0, 1) = \{q_5\}$

$\delta(q_6, 0) = \{q_6\}$ $\quad \delta(q_6, 1) = \{q_4\}$

The pair of states under '0' and '1' are

(q_1, q_6) are belongs to same subset and

(q_5, q_4) are belongs to same subset. so $q_0 \equiv q_6$.

so q_0 and q_6 are belongs to 1-equivalent class.

let $q_0, q_7 \in Q_2^0$

$\delta(q_0, 0) = \{q_1\}$ $\quad \delta(q_0, 1) = \{q_5\}$

$\delta(q_7, 0) = \{q_6\}$ $\quad \delta(q_7, 1) = \{q_2\}$.

The pair of states under '0' and '1' are

(q_1, q_6) are belongs to same subset

(q_5, q_2) are not belongs to same subset,

q_0, q_7 are not 1 equivalent class of each.

so, $[q_0, q_4, q_6]$ are belongs to 1-equivalent class.

let $q_1, q_5 \in Q_2^0$, then

$\delta(q_1,0) = \{q_6\}$ $\quad$ $\delta(q_1,1) = \{q_2\}$

$\delta(q_5,0) = \{q_2\}$ $\quad$ $\delta(q_5,1) = \{q_6\}$.

The pair of states under '0' and '1' are

(q_6, q_2) are not belongs to same subset, so

(q_2, q_6) are not belongs to same subsets. so,

$q_1 \neq q_5$. ie q_1 and q_5 are not in 1 – equivalence class.

Let $q_1, q_7 \in Q_2^0$, then

$\delta(q_1,0) = \{q_6\}$ $\quad$ $\delta(q_1,1) = \{q_2\}$

$\delta(q_7,0) = \{q_6\}$ $\quad$ $\delta(q_7,1) = \{q_2\}$.

The pair of states under '0' and '1' are

(q_6, q_6) are belongs to same subsets

(q_2, q_2) are belongs to same subsets

so, q_1 and q_7 are 1 – equivalent class

Then $\Pi_1 = \{\{q_2\}, \{q_0, q_4, q_6\}, \{q_1, q_7\}, \{q_5\}\}$.

Step 4: Construct Π_2. ie 2 – equivalence set'

$\Pi_1 = \{\{q_2\}, \{q_0, q_4, q_6\}, \{q_1, q_7\}, \{q_5\}\}$

Consider $Q_1' = \{q_2\}$, we can't partition Q_1'

Consider $Q_2' = \{q_0, q_4, q_6\}$

let $q_0, q_4 \in Q_2'$

then $(q_0,0) = \{q_6\}$ $\quad$ $(q_0,1) = \{q_5\}$

$(q_4,0) = \{q_7\}$ $\quad$ $(q_4,1) = \{q_5\}$,

The pair of states under '0' and '1' are

(q_1, q_7) are belongs to same subset

(q_5, q_5) are belongs to same subset. So, $q_1 = q_5$

ie q_1 and q_4 are two equivalent class.

Let $q_0, q_6 \in Q_2'$, then

$\delta(q_0, 0) = \{q_1\}$ $\delta(q_0, 1) = \{q_5\}$

$\delta(q_6, 0) = \{q_6\}$ $\delta(q_6, 1) = \{q_4\}$

Here, the pair of states under '0' and '1' are

(q_1, q_6) are belongs to two different subsets.

(q_5, q_4) are also belongs to two different subsets, so,

q_0 and q_6 are not two-equivalent class.

So, $[q_0, q_4]$ and $[q_6]$

Let $q_1, q_7 \in Q_3'$, then

$\delta(q_1, 0) = \{q_6\}$ $\delta(q_1, 1) = \{q_2\}$

$\delta(q_7, 0) = \{q_6\}$ $\delta(q_7, 1) = \{q_2\}$

The pair of states under '0' and '1' are

$\{q_6, q_6)$ are belongs to same subset

(q_2, q_2) are also belongs to same subset.

So, ~~$q_1 = q_7$~~ $q_1 = q_7$, ie q_1 and q_7 are two equivalent class.

Let $q_5 \in Q_4'$.

We can't partition it. So

The final $\Pi_2 = \{\{q_2\}, \{q_0, q_4\}, \{q_1, q_7\}, \{q_6\}, \{q_5\}\}$

Step 5: Construction of Π_3 i.e 3-equivalent class.

Let $\Pi_2 = \{\{q_2\}, \{q_0, q_4\}, \{q_1, q_7\}, \{q_6\}, \{q_5\}\}$

Then

Let $q_2 \in Q_1^2$, but q_2 is can't partition.

Let $q_0, q_4 \in Q_2^2$, then

$\delta(q_0, 0) = \{q_1\}$ $\quad \delta(q_0, 1) = \{q_5\}$

$\delta(q_4, 0) = \{q_7\}$ $\quad \delta(q_4, 1) = \{q_5\}$.

i.e $q_0 \equiv q_4$.

The pair of states under '0' and '1' are

(q_1, q_7) are belongs to same subset

(q_5, q_5) are belongs to same subset, so

q_0 and q_4 are 3-equivalent class.

Let $q_1, q_7 \in Q_3^2$, then

$\delta(q_1, 0) = \{q_6\}$ $\quad \delta(q_1, 1) = \{q_2\}$

$\delta(q_7, 0) = \{q_6\}$ $\quad \delta(q_7, 1) = \{q_2\}$

Then, the pair of states under '0' and '1' are $q_1 \equiv q_7$

(q_6, q_6) are belongs to same sub set

(q_2, q_2) are belongs to same sub set. Hence,

q_1 and q_7 are 3-equivalent subset.

Let $q_6 \in Q_4^2$, then q_6 can't partition

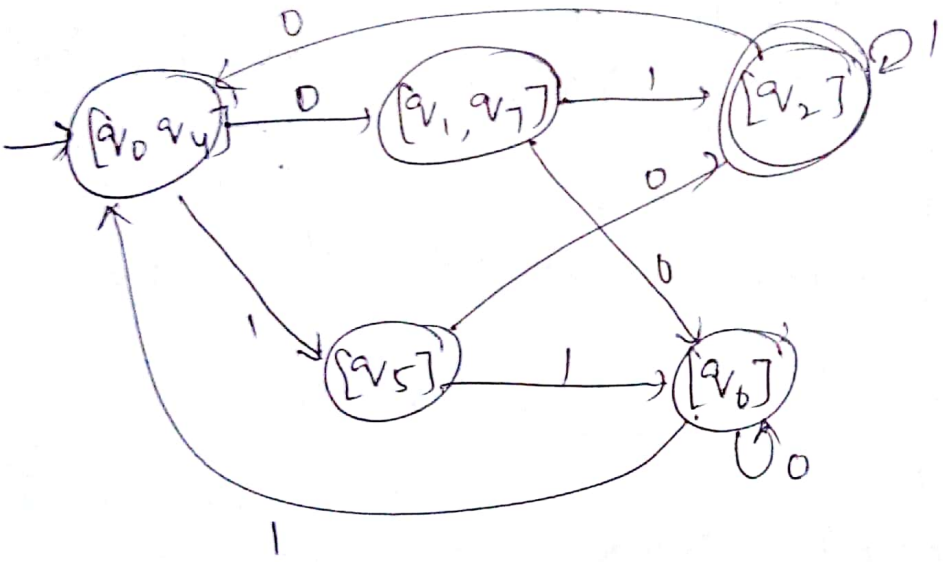
Let $q_5 \in Q_5^2$, then q_5 can't partition, then

$\Pi_3 = \{\{q_2\}, \{q_0, q_4\}, \{q_1, q_7\}, \{q_6\}, \{q_5\}\}$

If we observe, $\Pi_2 = \Pi_3$, then we stop the process. The states of required optimized DFA are equivalent classes of Π_3.

So, The final optimized DFA is shown in below fig.

$\Pi_3 = \{[q_0, q_4], [q_2], [q_1, q_7], [q_6], [q_5]\}$



	0	1
→ $[q_0, q_4]$	$[q_1, q_7]$	$[q_5]$
$[q_1, q_7]$	$[q_6]$	$[q_2]$
* $[q_2]$	$[q_0 q_4]$	$[q_2]$
$[q_5]$	$[q_2]$	$[q_6]$
$[q_6]$	$[q_6]$	$[q_0, q_4]$

Finite Automata with output:

- There are two finite Automata with outputs. They are
  - (i) Moore Machine
  - (ii) Mealy Machine.

Moore Machine:

- A moore machine is a finite state automaton, where the outputs are determined by the current state alone.
- A moore machine associates an output symbol with each state, and each time a state is entered, an output is obtained simultaneously. So, the first output always occurs as soon as the machine starts.

A moore machine is represented as a six tuple

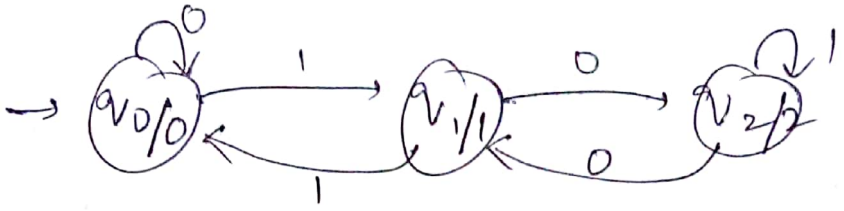
$M = \{Q, \Sigma, \Delta, \delta, \lambda, q_0\}$, where

Q: set of finite states
Σ: set of finite input symbols
Δ: set of finite output symbols
δ: transition function from $Q \times \Sigma \to Q$.
λ: o/p function which maps Q to Δ (ie $Q \to \Delta$)
q_0: An initial state.

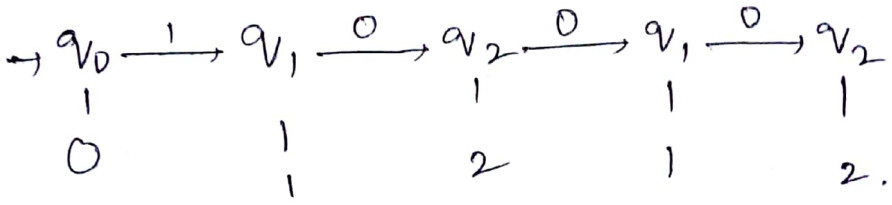
Ex: Draw the transition diagram for below transition table.

States	inputs		output
	0	1	Δ
→q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

Sol: The transition machine corresponding the given transition table is



input string w=1000



∴ the output for the above input '1000' is

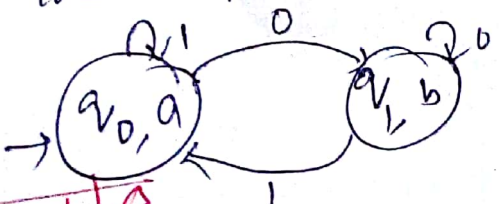
'0 1 2 1 2'

Input string |w| = n = 4
output string = n+1 = 5

Input = n
output = n+1

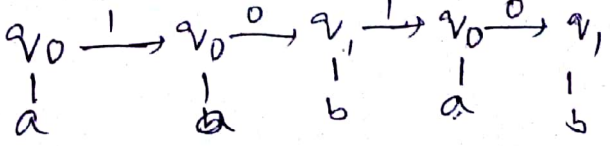
NOTE:
(i) There are no accept states in a moore machine because it is not a language recognizer but an output producer.
(ii) In a moore machine for the input string of length n, the output sequence consists of (n+1) symbols.

Ex:



	0	1	Δ
→q_0	q_1	q_0	a
q_1	q_1	q_0	b

(w) 1010 = (s) aabab



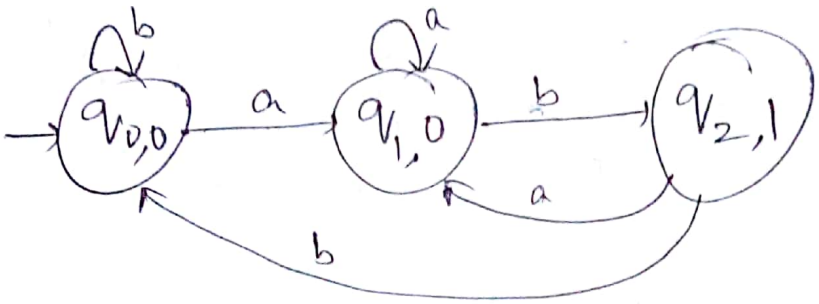
Ex:- Construct a moore machine that takes set of all the strings over {a,b} as input and prints '1' as the output for every occurance of 'ab' as a substring.

Sol:

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

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

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



Q \ Σ	a	b	Δ output
q_0	q_1	q_0	0
q_1	q_1	q_2	0
q_2	q_1	q_0	1

ab

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

$\lambda(\delta(q_0, ab)) = \lambda(\delta(\delta(q_0, a), b))$

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

$= \lambda(q_2)$

$= 1$

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2$

0 0 1

aab

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

$\lambda(\delta(q_0, aa)) = \lambda(\delta(\delta(q_0, a), a)$

$= \lambda(\delta(q_1, a))$

$= \lambda(q_1) = 0$

$\lambda(\delta(q_0, aab)) = \lambda(\delta(\delta(q_0, aa), b)$

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

$= \lambda(q_2) = 1$

$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2$

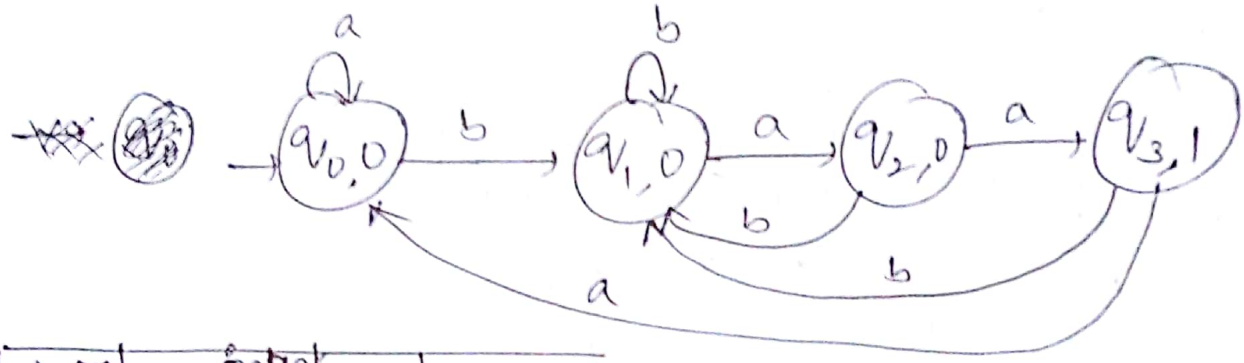
0 0 0 1

1) Construct a moore machine that takes set of all strings over {a,b} and counts no. of occurences of substring 'baa'

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

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

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



Q \ Σ	input a	input b	Δ output
q_0	q_0	q_1	0
q_1	q_2	q_1	0
q_2	q_3	q_1	0
q_3	q_0	q_1	1

Input string : baa

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

$\lambda(\delta(q_0, ba)) = \lambda(\delta(\delta(q_0, b), a))$

$= \lambda(\delta(q_1, a))$

$= \lambda(q_2) = 0$

$\lambda(\delta(q_0, baa)) = \lambda(\delta(\delta(q_0, ba), a))$

$= \lambda(\delta(q_2, a))$

$= \lambda(q_3) = 1$

input string : baa = 3 (n)

output : 0001 = 4 (n+1)

$q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_3$

1 1 1 1

0 0 0 1 ✓

Q.4 Construct a moore machine that takes set of all strings over {0,1} and produce 'A' as output if input ends with '10' or produce 'B' as output if input ends with '11' otherwise 'C'

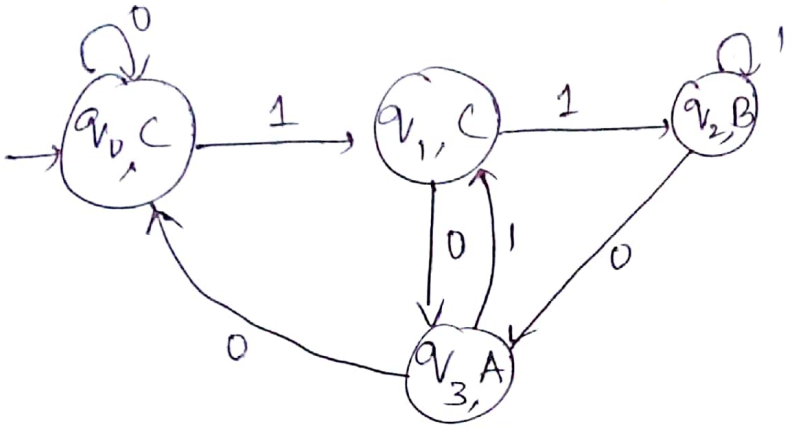
Sol: $M = \{Q, \Sigma, \delta, q_0, \Delta, \lambda\}$

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

$\Delta = \{A, B, C\}$

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

10 → A
11 → B
otherwise C



Q \ Σ	Input 0	Input 1	Output Δ
q_0	q_0	q_1	C
q_1	q_3	q_2	C
q_2	q_3	q_2	B
q_3	q_0	q_1	A.

111

$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \xrightarrow{1} q_2$

C C B B ✓

Input string : 111

output : C C B B ✓

$\lambda(\delta(q_0,1)) = \lambda(q_1) = C$

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

$\lambda(\delta(q_0,11)) = \lambda(\delta(\delta(q_0,1)),1)$

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

$= \lambda(q_2) = B$

$\lambda(\delta(q_0,111)) = \lambda(\delta(\delta(q_0,11)),1)$

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

$= \lambda(q_2) = B$ ✓

110

$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3$

C C B A ✓

for the following moore machine, the input alphabet is $\Sigma = \{a, b\}$ and the output alphabet is $\Delta = \{0, 1\}$. Run the following input sequences and find the respective outputs.

(i) aabab (ii) abbb (iii) ababb.

States	input a	input b	output (Δ)
→ q_0	q_1	q_2	0
q_1	q_2	q_3	0
q_2	q_3	q_4	1
q_3	q_4	q_4	0
q_4	q_0	q_0	0

Sol (i) aabab ⇒ output 001001

$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_4 \xrightarrow{a} q_0 \xrightarrow{b} q_2$

1 1 1 1 1 1

0 0 1 0 0 1

(ii) abbb ⇒ 00000

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_3 \xrightarrow{b} q_4 \xrightarrow{b} q_0$

1 1 1 1 1

0 0 0 0 0

(iii) ababb ⇒ 000001

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_3 \xrightarrow{a} q_4 \xrightarrow{b} q_0 \xrightarrow{b} q_2$

1 1 1 1 1 1

0 0 0 0 0 1

Mealy Machine

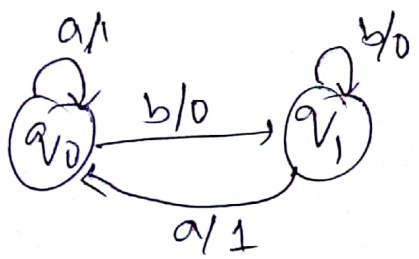
A Mealy machine is a finite state machine, where the outputs are determined by the current state and the input. A mealy machine associates an output symbol with each transition and the output depends on the current input.

The Mealy machine M is represented by 6 tuples

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

Q = finite set of states
Σ = input alphabets
δ = transition function. $Q \times \Sigma \rightarrow Q$
q_0 = initial state
Δ = finite set of output symbols
λ = output function. $\lambda: Q \times \Sigma \rightarrow \Delta$

Ex:-



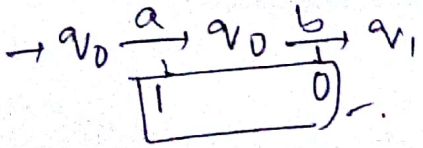
$M = (Q, \Sigma, \delta, q_0, \Delta, \lambda)$

Q	Input			
	a		b	
Σ	State	O/P(Δ)	State	O/P(Δ)
→ q_0	q_0	1	q_1	0
q_1	q_0	1	q_1	0

$\lambda(q_0, a) = 1$
$\lambda(q_0, b) = 0$
$\lambda(q_1, a) = 1$
$\lambda(q_1, b) = 0$

$\delta(q_0, a) = \{q_0\}$
$\delta(q_0, ab) = \delta(\delta(q_0, a), b)$
$= \delta(q_0, b) = \{q_1\}$

Input string ab



here the input.

Note:
- In a mealy machine for the input string of length n, the output sequence consists of n symbols not n+1
- There are no accept states in mealy machine because it is not a language recogniser, it is an output producer.

Ex: Construct a mealy machine to print out 1's complement of an input string.

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

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

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

input → 1010
o/p 1's comp 0101

State	Input 0	Input 1
	State o/p	State o/p
→ q_0	q_0 1	q_0 0

String: 1010

Input
State → $q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0$
o/p 0 1 0 1

output sequence

$\lambda(q_0, 1) = 0$

$\lambda(q_0, 0) = 1$

$\lambda(q_0, 1) = 0$

$\lambda(q_0, 0) = 1$

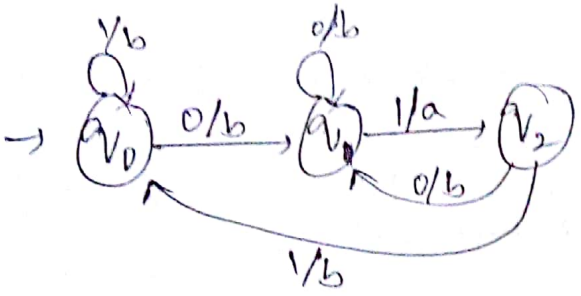
$\delta(q_0, 1010) = \delta(\delta(q_0, 1), 010)$
$= \delta(q_0, 010)$
$= \delta(\delta(q_0, 0), 10)$
$= \delta(q_0, 10)$
$= \delta(\delta(q_0, 1), 0)$
$= \delta(q_0, 0)$
$= \{q_0\}$

Ex: Construct a Mealy machine that prints 'a', whenever the sequence '01' is encountered in any input binary string.

Sol: $M = \{Q, \Sigma, \delta, v_0, \Delta, \lambda\}$

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

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



01

States	Input 0		Input 1	
	State	o/p	State	o/p
→ v_0	v_1	b	v_0	b
v_1	v_1	b	v_2	a
v_2	v_1	b	v_0	b

String: 01101

Input: 0 1 1 0 1

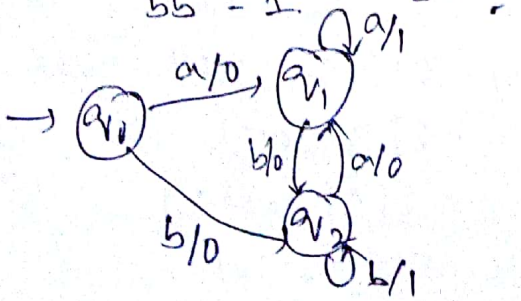
State: $v_0 \xrightarrow{0} v_1 \xrightarrow{1} v_2 \xrightarrow{1} v_0 \xrightarrow{0} v_1 \xrightarrow{1} v_2$

Output: b a b b a

$\delta(v_0, 01101) \vdash \delta(\delta(v_0, 0), 1101)$
$\vdash \delta(v_1, 1101)$
$\vdash \delta(\delta(v_1, 1), 101)$
$\vdash \delta(v_2, 101)$
$\vdash \delta(\delta(v_2, 1), 01)$
$\vdash \delta(v_0, 01)$
$\vdash \delta(\delta(v_0, 0), 1)$
$\vdash \delta(v_1, 1) \vdash \{v_2\}$

Ex: Design Mealy machine accepting the language consisting of string from Σ^*, where $\Sigma = \{a, b\}$ and the string should end with either aa or bb.

Sol: aa - 1 ab - 0
bb - 1 ba - 0



States	Input a		Input b	
	State	o/p	State	o/p
→ v_0	v_1	0	v_2	0
v_1	v_1	1	v_2	0
v_2	v_1	0	v_2	1

baa

Input: b a a

State: $v_0 \xrightarrow{b} v_2 \xrightarrow{a} v_1 \xrightarrow{a} v_1$

o/p: 0 0 1 ✓