Prof. Dipti Singh

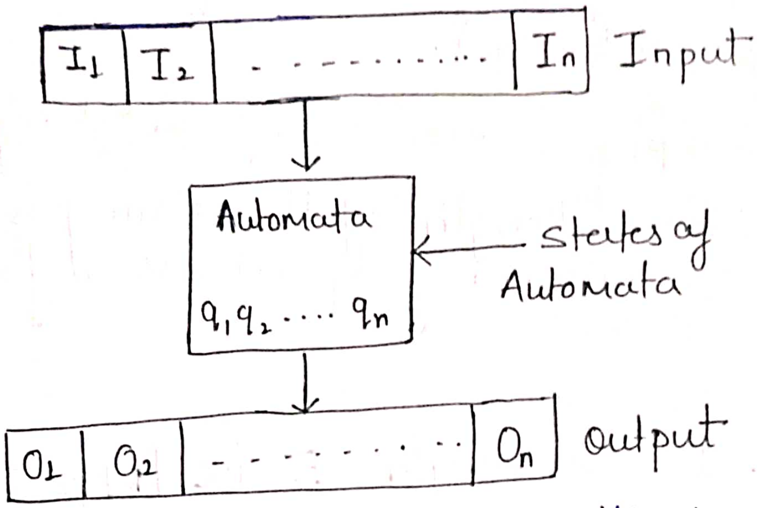
Unit 1

Theory of Automata & Formal Language

Introduction of Theory of Automata

Automata Theory is a branch of computer science that studies abstract machines and the logic of computation. It also studies the computational problems that can be solved using automata.

Objective :- To analyze the dynamic behavior of discrete systems



Simply stated, automata theory deals with the logic of computation with respect to simple machines, ~~refferred~~ referred to as automata.

A pioneer to Automata Theory [This is the Part of Unit 5]
(ALAN TURING (1912–1954)

- Father of Modern computer
- An English Mathematician
- Turing's Machine is essentially an abstract Model of Modern-day computer execution and storage.

Basic used in Theory of Computations:-

Everything in Mathematics is based on symbols. This is also true for ~~everything~~ automata theory of Computation. The symbols are generally letters, digits. The central concept of automata theory include the "alphabet" (a set of symbol) "string" (a list of symbols from an alphabet) and "language" (a set of strings from the same alphabet) (2)

eg → Binary symbols : $\Sigma = \{0,1\}$ ∴ Σ (Sigma)

All lower case letters : $\Sigma = \{a, b, c \ldots z\}$

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

Alphanumeric : $\Sigma = \{a-z, A-Z, 0-9\}$

Strings

A string or word is a finite collection of symbols selected from the alphabets (Σ).

A string can be empty, which is represented by ε ("Epsilon")

Length of String: The "Length" of the string is denoted by $|w|$ and it is the number of positions for the symbol in the string.

eg → w = 01101 has length = 5 i.e $|w| = 5$

ababaa length = 6 $|w| = 6$

Empty string

The "empty" string is the string with zero occurrence of Symbols. The empty string is represented by ε.

$\Sigma^* = \{\varepsilon, 0, 1, 01, 10, 000, 010, 0000 \ldots\}$ $\{* = 1, 2, 3 \ldots \infty\}$

$\{0\}^* = \{\varepsilon, 0, 00, 000, 0000 \ldots\}$

$\{1\}^* = \{\varepsilon, 1, 11, 111, 1111 \ldots\}$

Note that

ε is in Σ^*, regardless of what alphabet Σ is.

i.e ε is only string whose length is 0.

Concatenation of strings

Let x and y be two strings. Then xy denotes the concatenation of x and y, i.e, the string formed by making a copy of x and followed it by a copy of y. Example: if $x = ab$ & $y = cd$ the concatenation: $xy = abcd$. (Note - It is not commutative)

eg → $w = ab$, $x = ba$ = $wx = abba$

Reverse of the string

Reverse of the string can be achieved by simply interchanging over last symbols.

eg → $w = abc$, then $(w)^R = cba$.

Power of Alphabet :- Σ^*, $* = 0, 2, 3, \ldots \infty$

Let Σ be the alphabet. Then power of alphabet is given by

$$\Sigma^k = \text{the set of all strings of length } k.$$

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

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

Languages :- Language is a set string from the same alphabet. The lang. is undefined as similar to infinity which is undefined & this is similar to an ~~eg~~. empty box.

eg → $L_1 = \{ab, aabb, aaabbb\}$ is a lang ~~over~~ over alphabet $\Sigma = \{a, b\}$

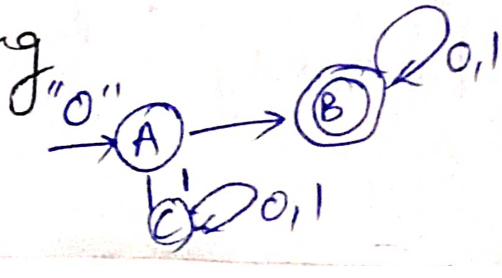
$L_2 = \{\varepsilon, a, aa, aaa, aaaa \ldots\}$ is a lang over alphabet $\Sigma = \{0, b\}$

$L_3 = \{a^n b^n c^n \mid n \geq 1\}$ is a lang.

$L_4 = \{a^n b^n c^m \mid m, n \geq 1\}$ is a lang.

L_1 = Set of all string that start with "0"

$\{0, 00, 01, 000, 010, 001, 0011, 0000$



L_5 = The set of string over 0's and 1's with equal no. of each (4)

$L = \{\varepsilon, 01, 1001, 0101, 011010 \ldots\}$

Note $\phi \neq \{\varepsilon\}$ ~~The former has~~ The former has no string & latter has one string

Substring :- Any string of consecutive symbols in some string 'w' can be collectively said as a ~~st~~ substring
eg→ w = abab its substring can be ab, a, ba .. etc.

* If a string has 'n' distinct symbols then total number of different sub string will be $[n(n+1)/2] + 1$.

What is Kleene closure Σ^* / Reflexive Transitive closure)

Q. If $\Sigma = \{a, b\}$ then

$\Sigma^0 = \{\varepsilon\}$

$\Sigma^1 = \{a, b\}$

$\Sigma^2 = \{aa, ab, ba, bb\}$

$\Sigma^3 = \{aaa, aab \ldots \ldots bbb\}$

* $\Sigma^K = \{W / |W| = K\}$ (using the symbols from the alphabet Σ)

Kleene closure:- If Σ is a set of symbols, then we use Σ^* to denote the set of strings obtained by concatenating zero or more symbols from Σ of any length, in general any string of any string ~~of~~ of any ~~to~~ length which can have only symbols specified in Σ.

$\Sigma^* = \cup_{i=0}^{i=\infty} \{w \,/\, |w| = i\}$ (using the symbols from the alphabet Σ}

eg →

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

① If $\Sigma = \{0,1\}$ then $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$

$\Sigma^0 = \{\varepsilon\}$

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

$\Sigma^2 = \{00, 01, 10, 11\}$ and so on

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

② If $s = \{a\}$ then

$\Sigma^* = \{\varepsilon, a, aa, aaa, aaaa, \ldots\}$

<u>Positive closure</u> :- If Σ is a set of symbols, then we use Σ^+ to denote the set of strings obtained by concatenating one or more symbols from Σ of any length, in general any string of any length which can have only symbols specified in Σ (except ε).

$\Sigma^+ = \cup_{i=1}^{i=\infty} \{w \,/\, |w| = i\}$ (using the symbols from the alphabet Σ)

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

Q.) Given the language $L = \{ab, aa, baa\}$ which of the following strings are in L^*

$L^* = \{\varepsilon, ab, aa, baa, abaa, aabaa\}$

difference Σ^* or L^*

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

$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb \ldots\}$

6

Q) The number of substring 'adefbghmmp"

$\frac{n(n+1)}{2} + 1$ — 1 for ε

Q) Suppose $L_1 = \{10, 1\}$ and $L_2 = \{011, 11\}$
How many elements are there in $L = L_1 L_2$

= 3 An.

$w_1 w_2$

$\{10, 1\}$ $\{011, 11\}$

$\{10011, 1011, 1011, 111\}$

$\underbrace{1011, 1011}_{\times}$

Grammar

not rescricted

type 3 grammar / RG (high) Regular grammar → RL Regular languange → Finite Automata (

CFG (less high) → CFL → PDA (Push Down automata)

CSG (Context sensitive grammar) → CSL Contect Sensitive Language → LBA Linar bounded automata

REG (Recursive enumerable ~~language~~ grammar) → REL Recursive enumerable language) → TM Turing Machine

language ko accept karega jo vo hi finite automata

Formal language :-

To define the concept of a formal lang. we need the idea of an alphabet and string we have seen the Kleene clausure & positive closure. A formal Language L on the alphabet Σ is a subset of Σ^* i.e $L \subseteq \Sigma^*$. Here we are defining a formal lang. L as a collection of string (or words)

GRAMMAR

Grammar is nothing but set of rules used to define the lang. These rules can be written with the help of terminal or non-terminal

eg →

$G = \{V \; T \; P \; S\}$

Variable, terminal, Production Rule, start

eg → ① $S \longrightarrow aSb / \varepsilon$

ε, aSb, $aaSbb$, $aaaSbbb$ ... $a^n b^n$

a^2Sb^2, a^3Sb^3, a^4b^4 ... $n \geq 0$

eg → ②

$S \rightarrow SS$

$S \rightarrow aSb$

$S \rightarrow bSa$

$S \rightarrow \varepsilon$

$L = \{\varepsilon$, $aSbbSa$, $absabasb$
abba, ababab

Language → $n_a(w) = n_b(w)$

eg → 3 $S \longrightarrow 1S / 0S$

$S \longrightarrow 0/1$

terminal symbols are 0 and 1

$S \rightarrow 1S$
$\rightarrow 10S$
$\rightarrow 101S$
$\rightarrow 1011$

$L = \{0, 1, 00, 11, 10, 01, 1010, 1011 \ldots\}$

Finite automaton/automata

A Finite automaton is a model that has a finite set of states. (~~represented in~~ represented in the figure by circles) and its control moves from one state to another state in response to ~~an~~ external inputs (represented by arrows)

Finite automata can be broadly classified into two types.

1. Finite automata without ~~op~~ output
   1. Deterministic finite automata.
   2. Non-deterministic finite automata
   3. Non-deterministic finite automata with ε

2. Finite automata with output.
   1. Moore Machine
   2. Mealy Machine

Deterministic Finite Automata

A deterministic finite automata (DFA) is defined by 5-~~tag~~ tuple $(Q, \Sigma, \delta, S, F)$ where

Q = Q is a finite and non-empty set of states

Σ = Σ is a finite non-empty set of finite input alphabet.

δ = δ is a transition function $(\delta : Q \times \Sigma \rightarrow Q)$

$q_0, a \rightarrow q_1$

S = S is initial state (always one) $(S \in Q)$ q_0

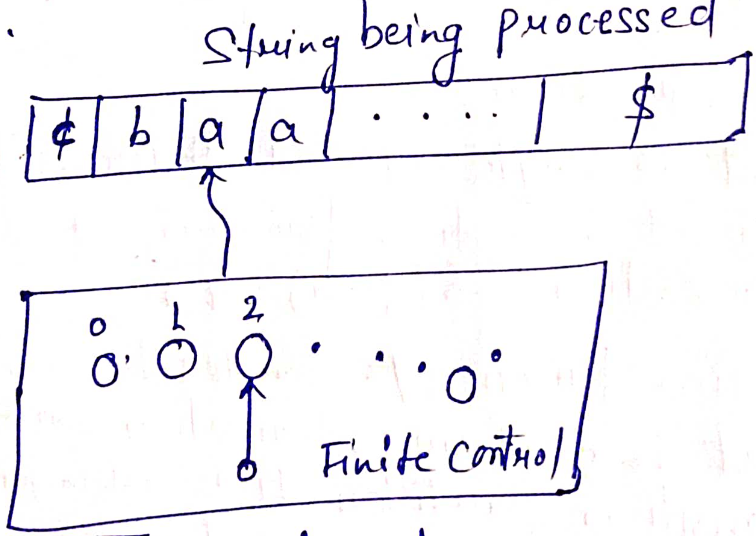
F = F is a set of final states $(F \subseteq Q)$ $(0 \leq |F| \leq N$, where n is the no. of states)

The various Components of the block diagram are explained as follows:

Input tape :- The input tape is divided into squares, each Square containing a single symbol from the input alphabet Σ. The end squares of the tape contain the end Marker ¢ at the left end and the end Marker \$ at the right end. The absence of end Markers indicates that the tape is of infinite length. The left to right sequence of symbols between the two end Markers is the input string to be processed.

• Reading head (R-head) The head examines only one square at a time and will move one square to the right.

Finite control: Is the inference engine take care of transition.



Finite Automaton

Note

- Produces a unique ~~computation~~ computation (or run) of the automaton for each input string.
- Deterministic refers to the uniqueness of the Computation.
- DFA are useful for doing lexical analysis (spell check) in compiler design.

Transition function

We have, two types of transition function based on the input. If length of the input is zero or one then direct transition otherwise indirect transition

Transition Function

- Direct transition function (represented by δ)
- Indirect transition funt (represented by δ' or $\hat{\delta}$)

Direct Transition function:

When the input is single symbol, ~~A~~ transition func is known as direct transition function. It is also known as one step transition ($\delta(p, a) = q$)

Indirect Transition function (Extended transition function)

When the input is a string, transition func is known as indirect transition function. It is also known as one step or more than one transition function. $\delta(p, w) = q$, ~~next~~ where p is the present state & q is the next state after transition.

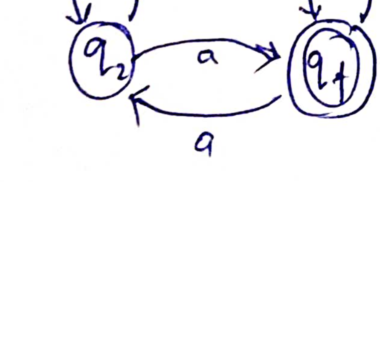
eg → $w = abb$

$\delta(q_0, aab) = \delta(\delta(q_0, a), ab)$
$= \delta(q_1, ab)$
$= \delta(\cancel{q}\,\delta(q_0, a), b)$
$= \delta(q_2, b)$
$= q_3$
$\cancel{= \delta(\delta(q_0, aab)}$,
$= \delta(q_0, aab) = q_3$

Transition Table

A transition table is a tabular representation of a transition function that takes two arguments and returns a state. The rows of the table corresponds to ~~that~~ the states and the columns corresponds to the inputs. The entry for one row corresponding to state q and the column corresponding to input 'a' is the state $\delta(q, a)$

eg → A DFA $M = (\{q_0, q_1, q_2, q_f\}, \{a, b\}, \delta, q_0, \{q_f\})$ is shown in figure. The graph shown in fig. is known as transition graph.



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

Or

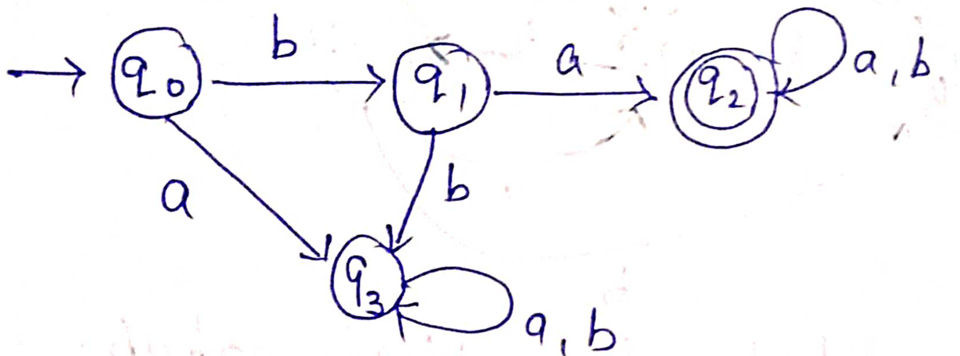
Q) What are limitations & Applications of FA?

[do it yourself]

13

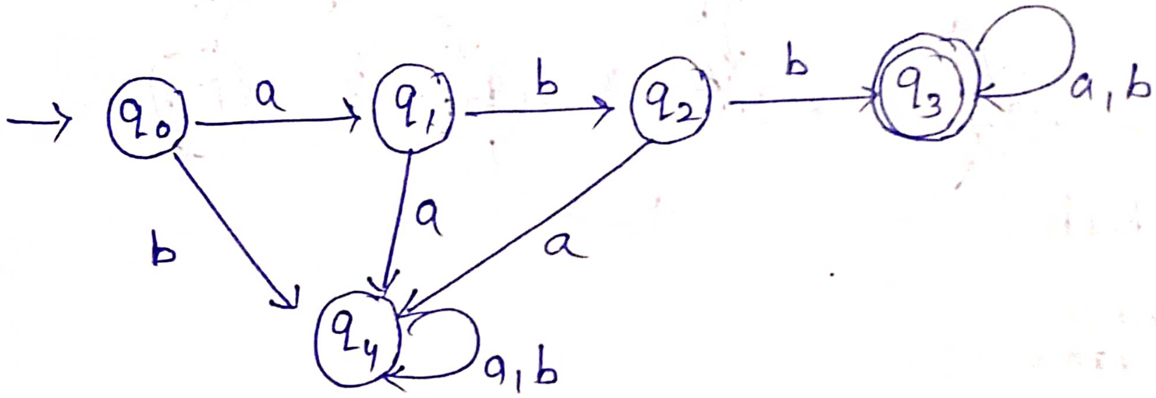
Q) Design a MDFA over $\Sigma = \{a, b\}$ such that every string accepted must start with a string w

w = ba L = { ba, baa, baba, babaa ... }



{ba no. of states = 2+2=4}

W = abb L = {abb, abba, abbb, abbaa ... }



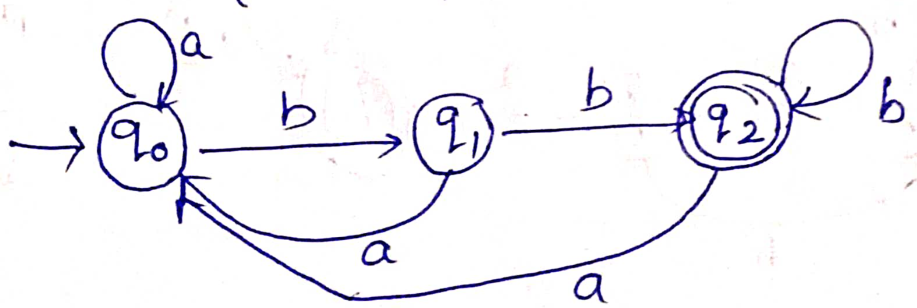
Q) Design a MDFA over $\Sigma = \{a, b\}$ such that every string accepted must ends with a substring w.

① W = bb

② W = ab

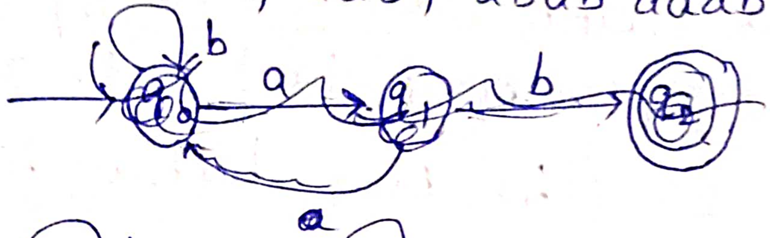
③ W = bab

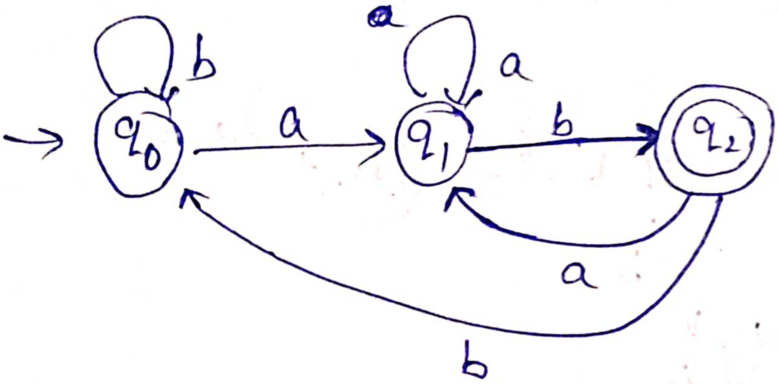
① W = bb L = { bb, abb, abbb, bbb, babb ... }



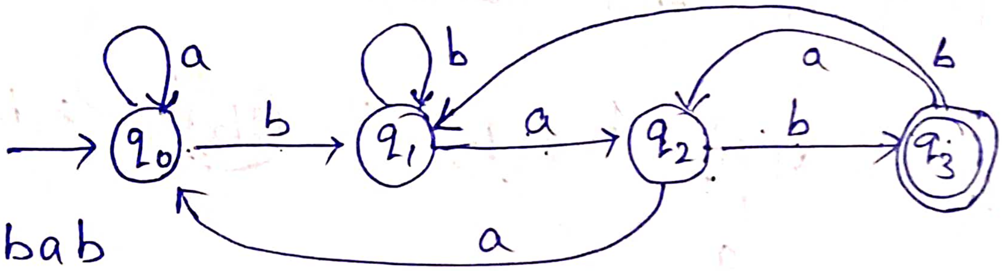
(2) $W = ab$ $L = \{ab, aab, abab\ aaab \ldots\}$

aaa
xxxab
bbbab





(3) $W = bab$ $L = \{bab, abab, bbab, aabab \ldots\}$

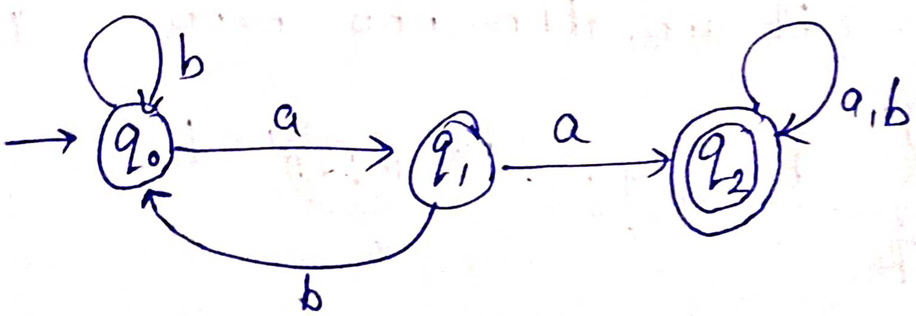


xxx bab
aaa

aaabbab
baa x
bab
aaabbab bab
bababa

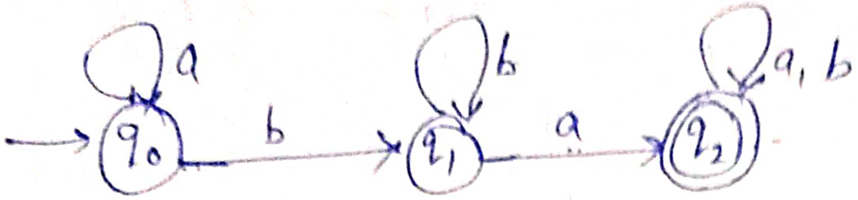
Q) Design DFA over $\Sigma = \{a, b\}$ such that every string accepted must contains a substring w.

$W = aa$ $L = \{aa, aaa, baa, aab, baab \ldots\}$



bbbaaa
abaa x
bbaabbaa

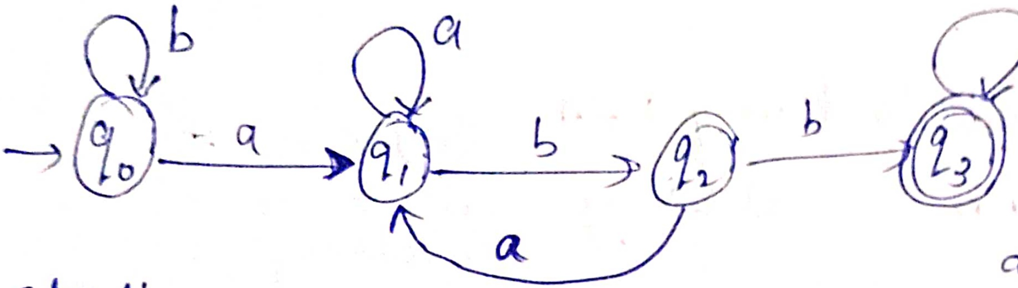
w = ba L = { ba, aba, bba, baa, bab ... }



baaa

babbb

w = abb L = { abb, aabb, abba, abbb ... }



abab bb

abaX

abab

aaabb

bbabb

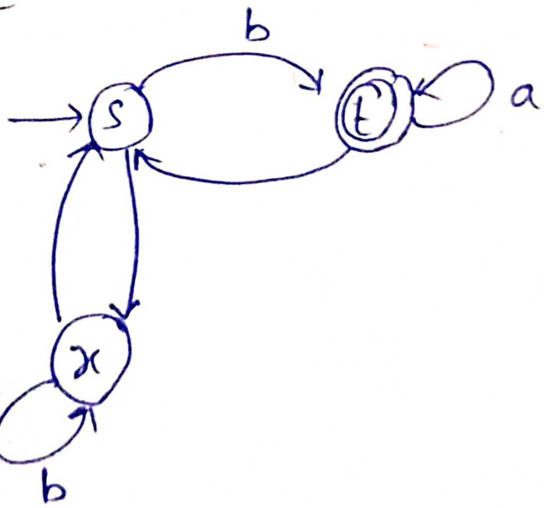
aaaabb

Q. In the automaton below s is the state and t are the only final state C

u = ababab

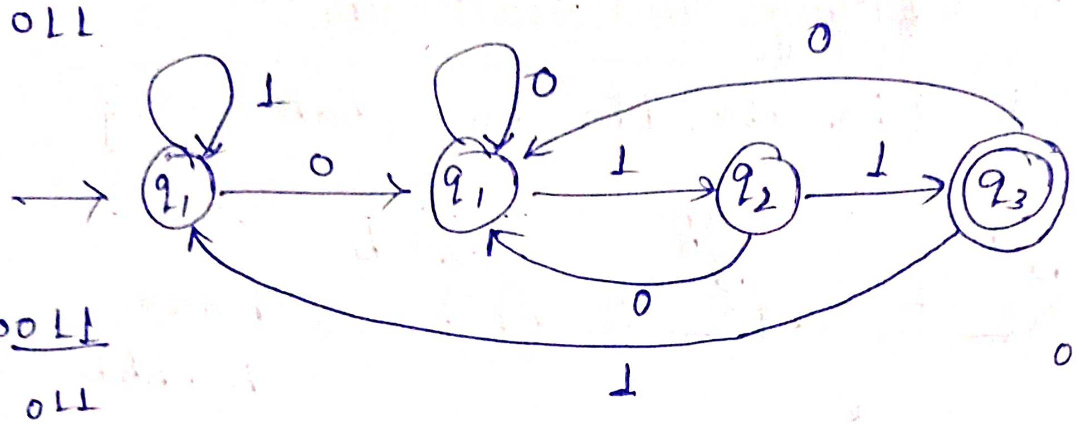
v = bab

w = aabb



$L = \{w \in \{0,1\}^* \mid w$ ends with the substring $011\}$ (14)

$w = 011$



000 11

011

11101

000 11

0 01011

01111

011

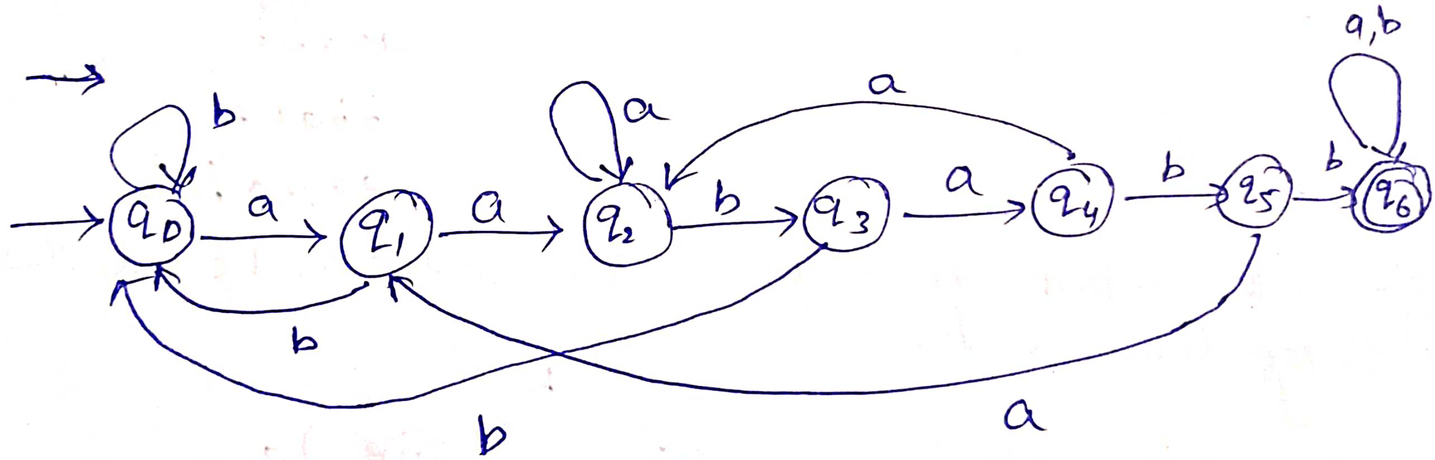
Q) Consider the following DFA

aababb — substring

Prefix ✗

Suffix ✗

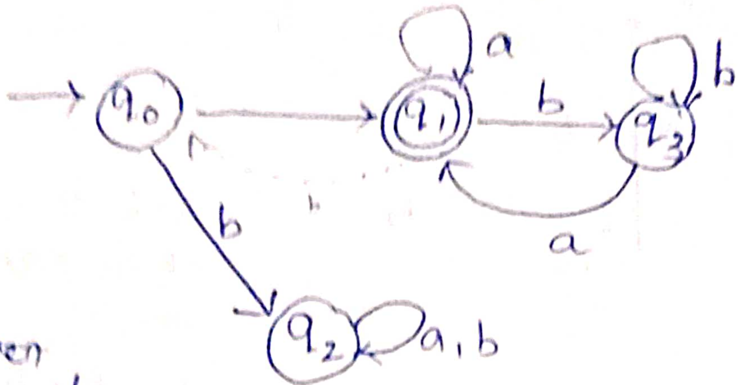
substring ✓



aababb

Q) Design a MDFA over $\Sigma = \{a, b\}$ such that every string accepted must start and ends with 'a' (15)

$L = \{a, aaa, aba\ abba, \ldots\}$



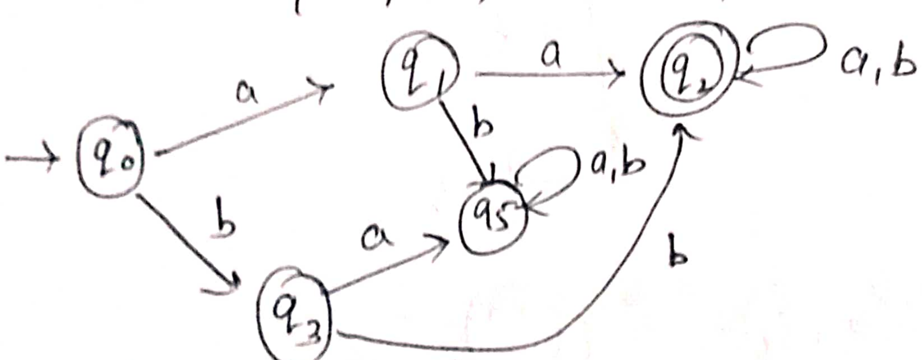
abba ✓

abb

Whenever ~~you can't~~ ~~... q_3~~, start return back to dead state start with that state.

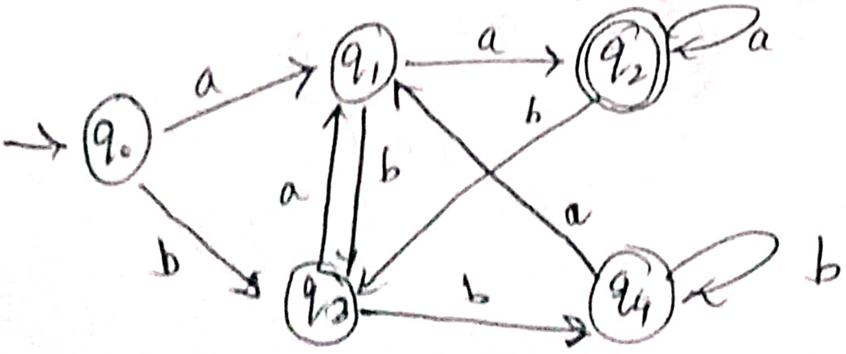
Q) Construct DFA over $\Sigma = \{a, b\}$ ~~over~~ such that string accepted must start with 'aa' or 'bb'

$L = \{aa, bb, aaxxx.., bbxxx\ldots\}$



must end with 'aa' or 'bb'

$L = \{aa, bb, xxxaa, xxxbb \ldots\}$



(18)

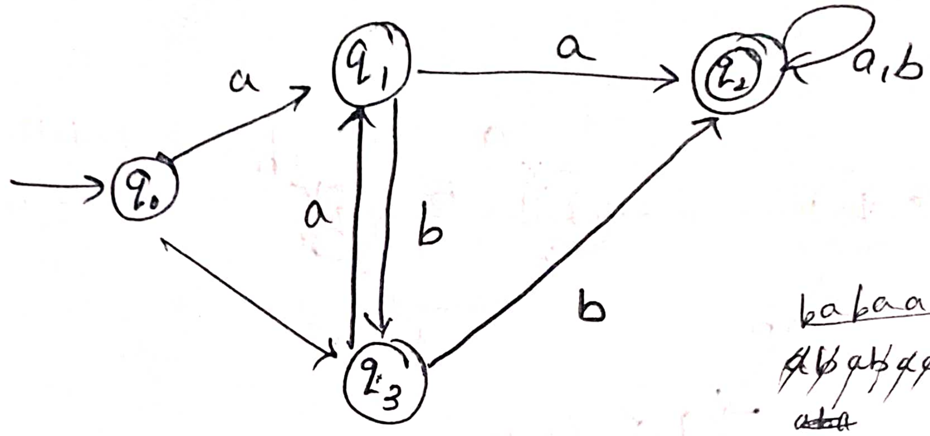
Design a MDFA over $\Sigma = \{a, b\}$ such that every string accepted must contains a substring aa or bb. (16)

xxaaxx xxbbxx

suffix → last
prefix — start
string → mid

$L = \{$ a|

aa
abb
baa



ababaaba

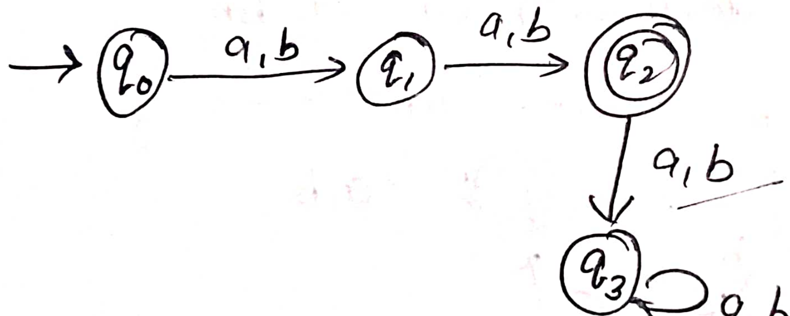
babaabab

~~abababaabb~~

~~aba~~

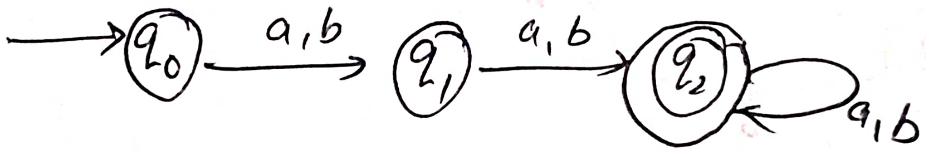
Q) Design a MDFA over $\Sigma = \{a, b\}$ such that string accepted must
(a) $|w| = 2$, $|w| = n$ $(n+2)$
(b) $|w| \geq 2$ $(n+1)$
(c) $|w| \leq 2$ $(n+2)$

(a)

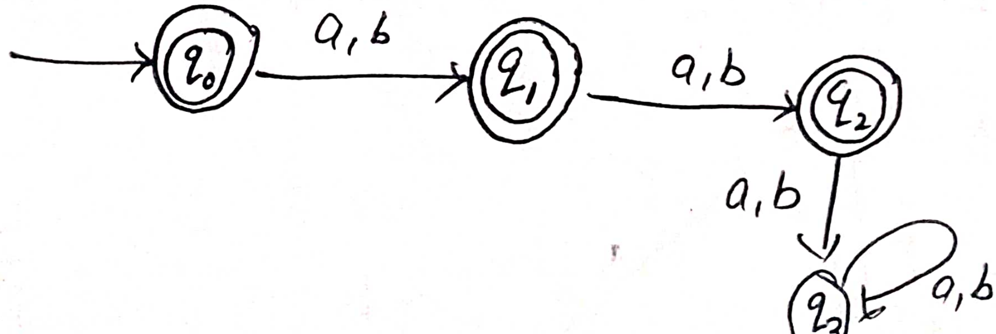


abab

(b)



(c)



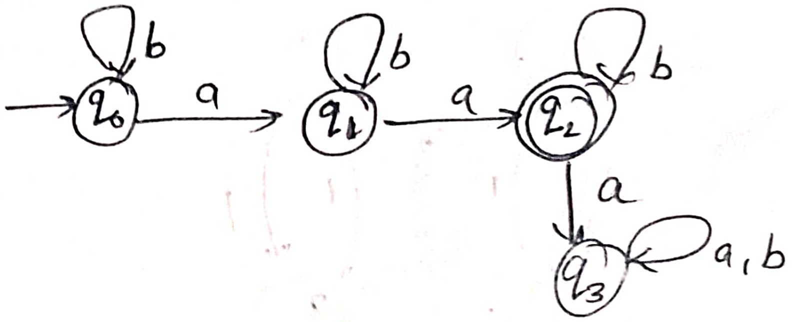
Design a MDFA over $\Sigma = \{a, b\}$ such that every string accepted must

(a) $|w|_a = 2$

(b) $|w|_a \geq 2$

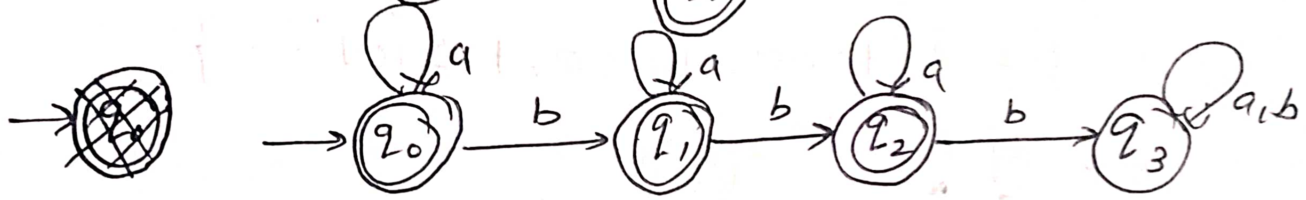
(c) $|w|_b \leq 2$

(a)



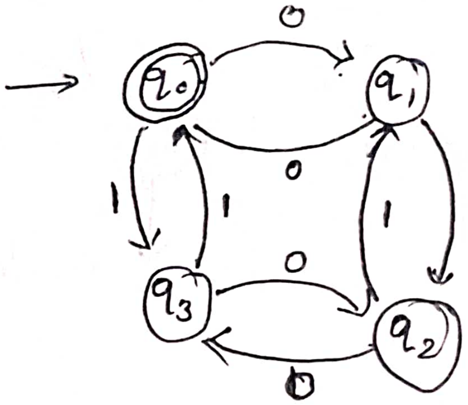
(b)

(c)

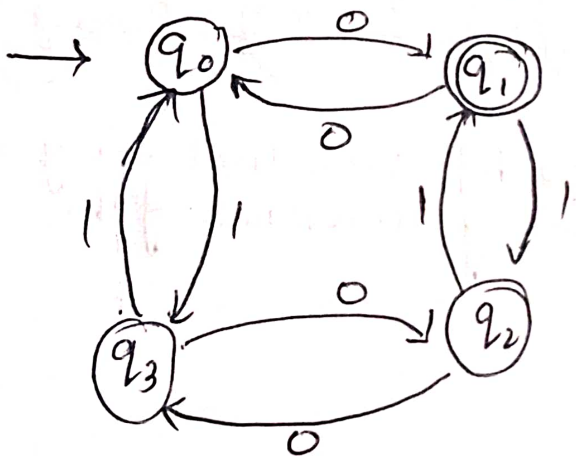


Construct a DFA for the following over alphabet (18)
$\Sigma = \{0,1\}$.

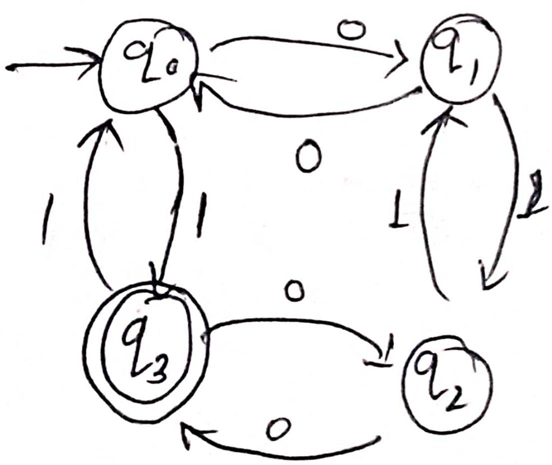
① Even no. of 1's and Even no. of 0's.
L = {0101, 1010, 101010 ... }



② Even no. of 1's and odd no. of 0's
L = {11000, 10100, 110101 ...}



③ odd no. of 1's and Even no. of 0's
L = {001111, 0101100 ...}



Extended Transition function

extended transition function $\delta^* : Q \times \Sigma^* \rightarrow Q$. The second argument of δ^* is a string, rather than a single symbol, and its value gives the state the automaton will be in after reading that string.

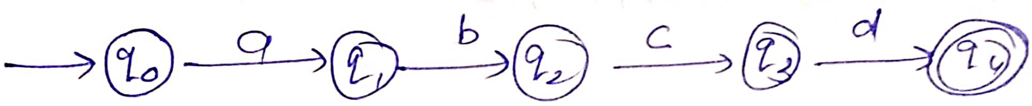
$$\delta(q_0, a) = q_1 \text{ and } \delta(q_1, b) = q_2$$

then $\delta^*(q_0, ab) = q_2$

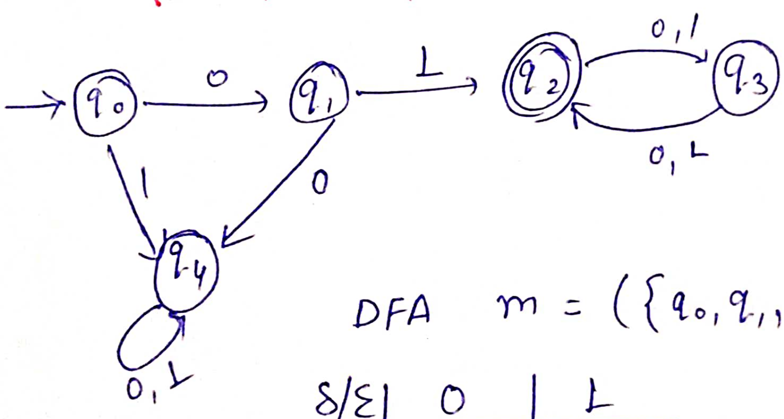
eg → string take abcd normal DFA

$\delta : (q_0, a) = q_1$
$\delta : (q_1 ; b) = q_2$
$\delta : (q_2 ; c) = q_3$
$\delta : (q_3 , c) = q_4$

$\delta^*(q_0, abcd) = q_4$



eg → Design a DFA to accept the length
$L = \{ w | w$ is a even length and begins with 01 $\}$
$L = \{ 01, 0111, 0110, 0100, 011100 \cdots \}$



DFA $m = (\{q_0, q_1, q_2, q_3, q_4\} \{0,1\}. \delta. q_0, \{q_2\})$

δ/Σ	0	1
$\rightarrow q_0$	q_1	q_4
q_1	q_4	q_2
q_2	q_3	q_3
q_3	q_2	q_2
q_4	q_4	q_4

~~const~~

Consider $w = 011101$

As w starts with 01 & it is of even length

$\therefore w \in L$ & $\hat{\delta}(q_0, 011101) = q_2$

check by computing $\hat{\delta}(q_0, w)$ for each prefix of $w = 011101$, string at ϵ

$$\hat{\delta}(q_0, \epsilon) = q_0$$

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

$$\hat{\delta}(q_0, 01) = \delta(\hat{\delta}(q_0, 0), 1) = \delta(q_1, 1) = q_2$$

$$\hat{\delta}(q_0, 011) = \delta(\hat{\delta}(q_0, 01), 1) = \delta(q_2, 1) = q_3$$

$$\hat{\delta}(q_0, 0111) = \delta(\hat{\delta}(q_0, 011), 1) = \delta(q_3, 1) = q_2$$

$$\hat{\delta}(q_0, 01110) = \delta(\hat{\delta}(q_0, 0111), 0) = \delta(q_2, 0) = q_3$$

$$\hat{\delta}(q_0, 011101) = \delta(\hat{\delta}(q_0, 01110), 1) = \delta(q_3, 1) = q_2$$

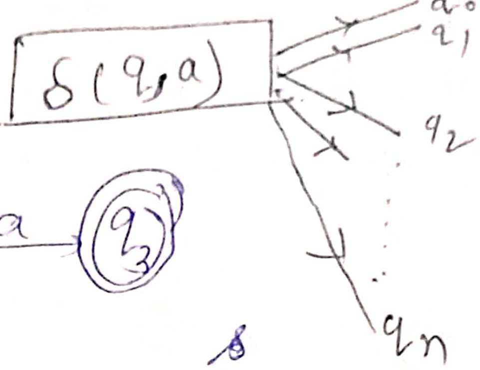
Non-Deterministic Finite Automaton

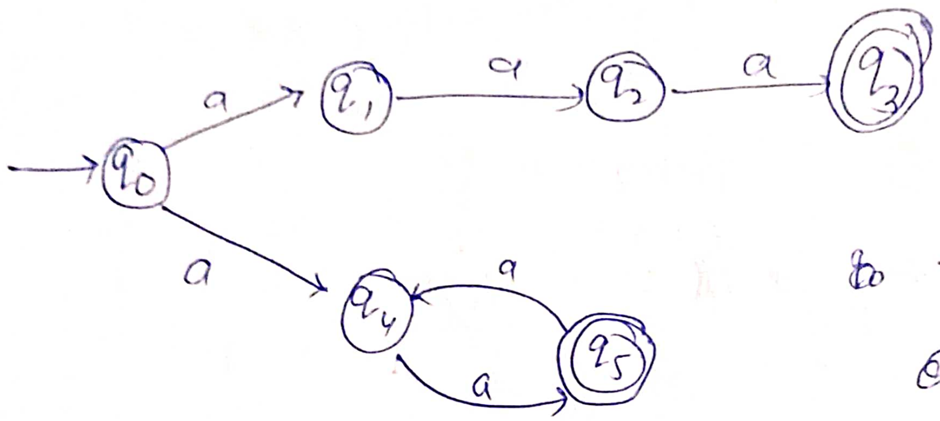
A nondeterministic finite accepter or nfa is defined by the quintuple

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

Where Q, Σ, q_0, F are defined as for deterministic finite accepters, but

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^{Q}$$





δ Totally working on power set

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

$|Q| = 3$

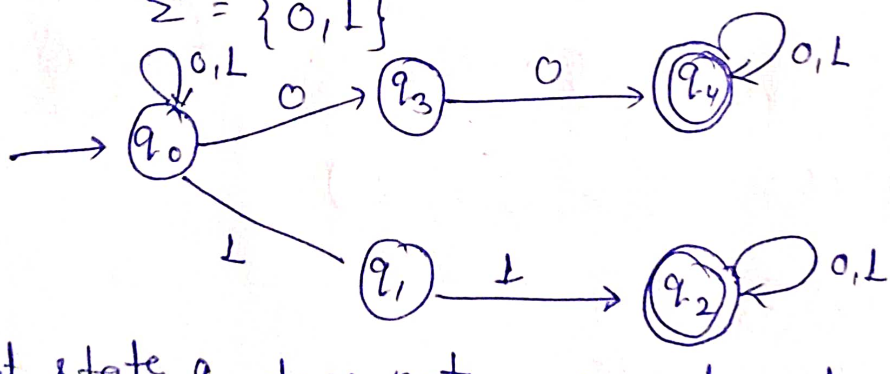
~~eg → Construct NFA for exactly 010 string~~

eg → Design a NFA for the language L = all string over $\{0,1\}$ that have at least two consecutive 0's or 1's

$L = \{00, 11, 101000, 101100 \ldots$

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

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



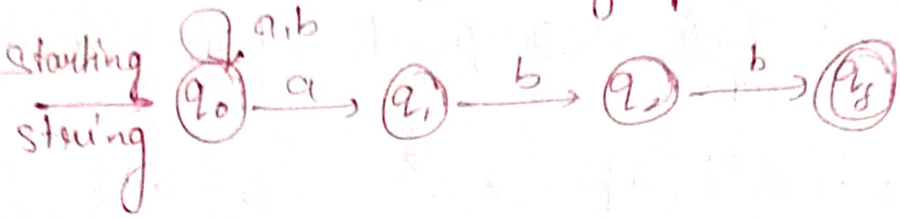
Note that state q_3 does not go anywhere for an input of 1. We use the terminology that path "dies." if in q_3 getting an input 1.

Acceptibility of string by NFA:

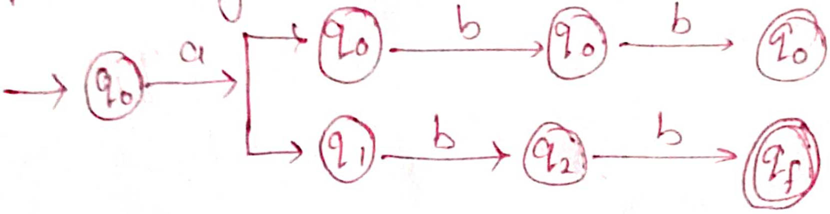
$$\delta(q_0, w) = q_f$$

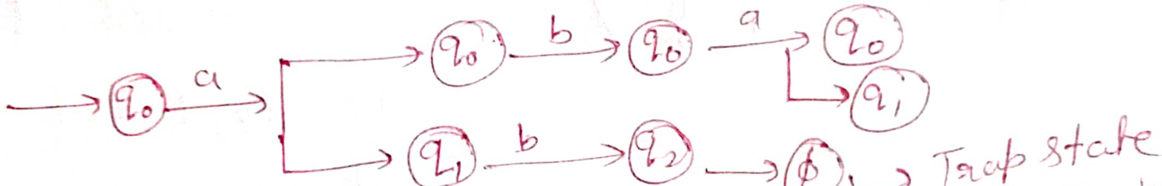
for example: check the acceptibility of following string

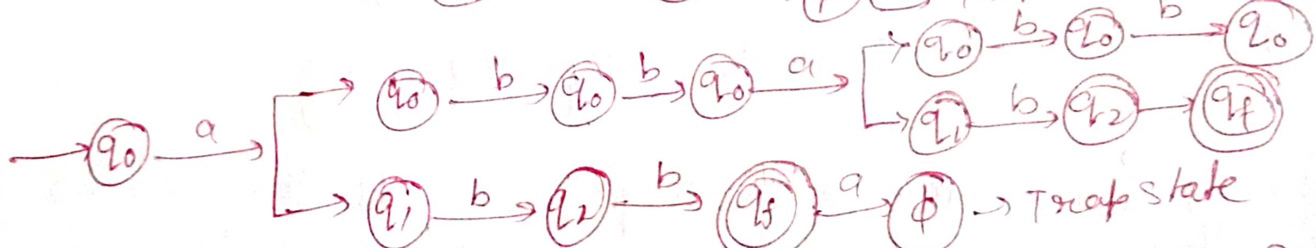
ⓐ abb
ⓑ aba
ⓒ abbabb



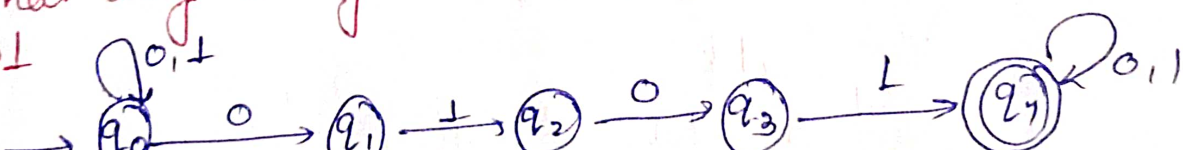
① Let w = abb then the transmission sequence for the input string "abb" is a follows:

ⓐ


ⓑ


ⓒ


Q) Design a NFA for the lang. L over alphabet $\Sigma = \{0, 1\}$ Suchthat every string is the L must contain the sub-string 0101



Sol Let NFA $M_1 = (Q, \Sigma, \delta, F, q_0)$

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

$q_0 = \{q_0\}$ initial state

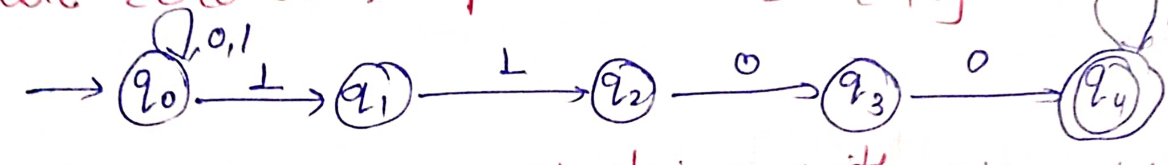
$F = \{q_4\}$ Final state

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

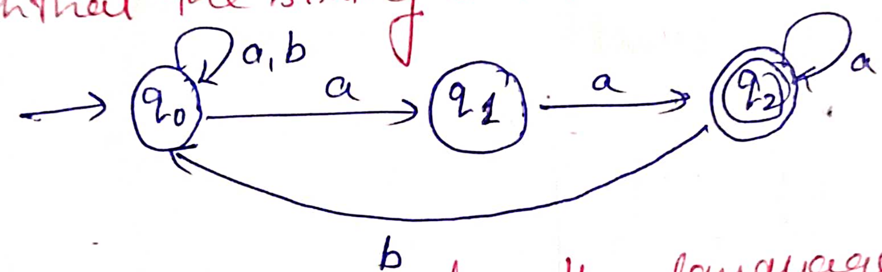
$\delta \mid \Sigma$	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	—	$\{q_2\}$
q_2	q_3	—
q_3	—	q_4
$*q_4$	q_4	q_4

Q) Construct a NFA in which double one is followed by double zero over alphabet $\Sigma = \{0,1\}$.

sol



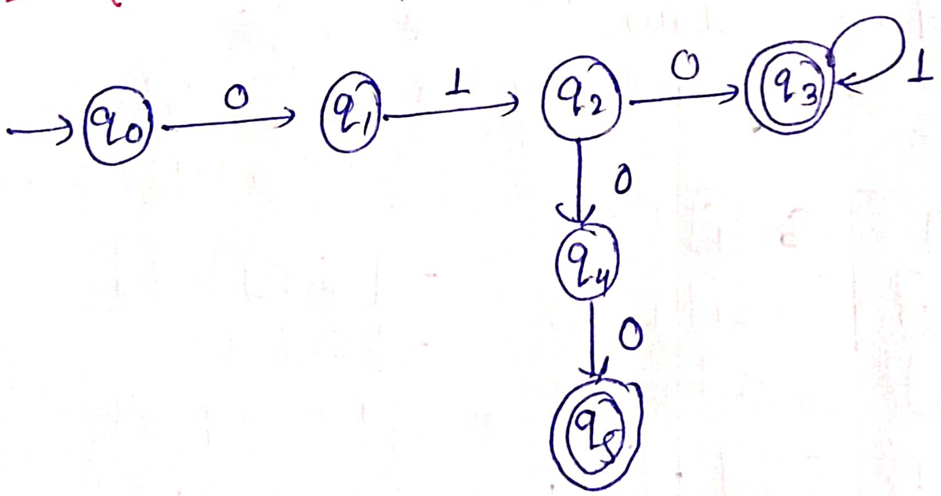
Q) Design a NFA to accept strings with a's and b's such that the string end with aa.



Q) Construct NFA for the language.

$L = \{0101^n \cup 0100 / n \geq 0\}$

$L = \{010, 0101, 01011, 010111 \cup 0100\}$



Q) Difference b/w DFA and NFA

{Do it yourself}

Conversion of NFA to DFA

Construction of Equivalent DFA M from given NFA M_1

Q) Consider a NFA shown in fig.

$[S = q_0]$

Sol Let given NFA, $M = (Q, \Sigma, \delta, S, F)$
and its equivalent DFA, $M_1 = \{Q_1, \Sigma, \delta, [S]\ F_1\}$

$Q = \{S, q_1, q_2, q_f\}$

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

$S = \{S\}$, starting state

$F = \{q_f\}$, final state

δ is defined as follows:

δ / Σ	a	b
$\rightarrow S$	$\{S, q_1\}$	$\{S\}$
q_1	—	$\{q_2\}$
q_2	—	$\{q_f\}$
$* q_f$	—	—

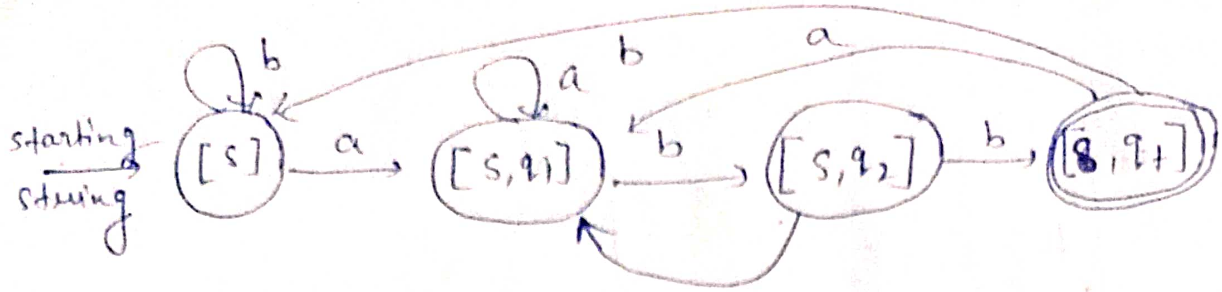
δ_1 is defined as follows:

δ_1 / Σ	a	
$\rightarrow [S]$	$[S, q_1]$	$[S]$
$[S, q_1]$	$[S, q_1]$	$[S, q_2]$
$[S, q_2]$	$[S, q_1]$	$[S, q_f]$
$* [S, q_f]$	$[S, q_1]$	$[S]$

~~* δ_1 [[S~~

$* \ \delta_1 [[S, q_1], a] =$
$[\delta(S, a) \cup \delta(q_1, a)]$
$= [S, q_1] \cup \phi]$
$= [S, q_1]$

$* \ \delta_1 [[S, q_1], b] =$
$[\delta[S, b] \cup \delta[q_1, b]]$
$= [\{S\} \cup \{q_2\}]$
$= [S, q_2]$



$\delta = \{[s]\}$ starting state $\quad F_1 = \{[S, q_f]\}$

$Q_1 = \{[s], [s, q_1][s, q_2][s, q_f]\}$

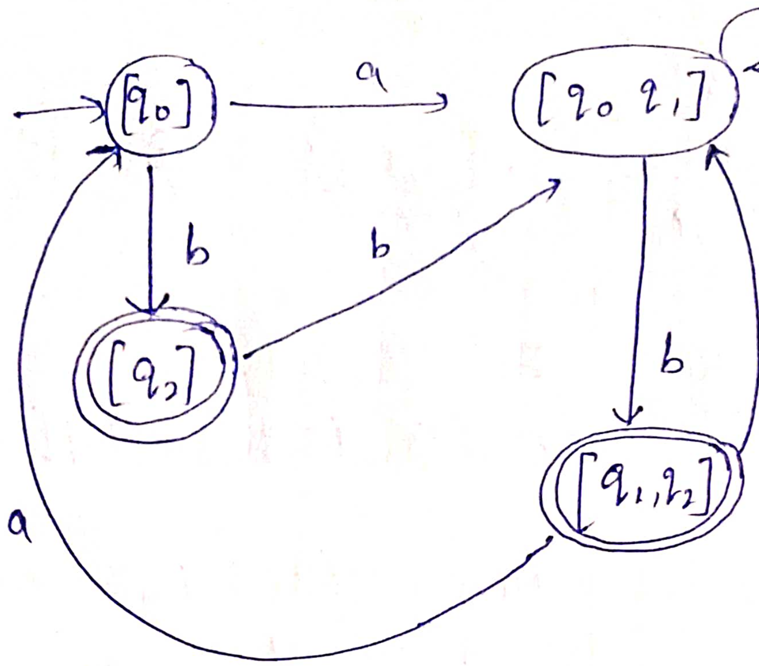
$\Sigma = \{a, b\}$

Q) Find a DFA equivalent to NFA $M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$ where δ is defined as follows –

present state (PS)	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_2\}$
q_1	$\{q_0\}$	$\{q_1\}$
$* q_2$	—	$\{q_0, q_1\}$

Sol Let $M_1 = (Q, \Sigma, \delta, q_0, F)$ be the equivalent DFA, where $\Sigma = \{a, b\}$, $q_0 = [q_0]$ starting state δ_1 is defined as follows:-

Present state (PS)	a	b
$\rightarrow [q_0]$	$[q_0, q_1]$	$[q_2]$
$[q_0 q_1]$	$[q_0, q_1]$	$[q_1, q_2]$
$* [q_2]$	—	$[q_0, q_1]$
$* [q_1, q_2]$	$[q_0]$	$[q_0 q_1]$



$Q_1 = \{ [q_0], [q_0, q_1], [q_2], [q_1, q_2] \}$

$F_1 = \{ [q_2], [q_1, q_2] \}$

Minimization of DFA.

Algorithm

① Eliminate the dead state and unreachable states from the given DFA [if any]

② Draw the transition table for rest states after removing the dead states and unreachable states.

③ Split the transition table in two tables T_1 and T_2, where

T_1 = { contains all rows which starts with states from Q - F }

T_2 = { contains all rows which starts with states from F }

④ Find the similar rows from T_1 such that:

$\delta(q_g, a) = P$

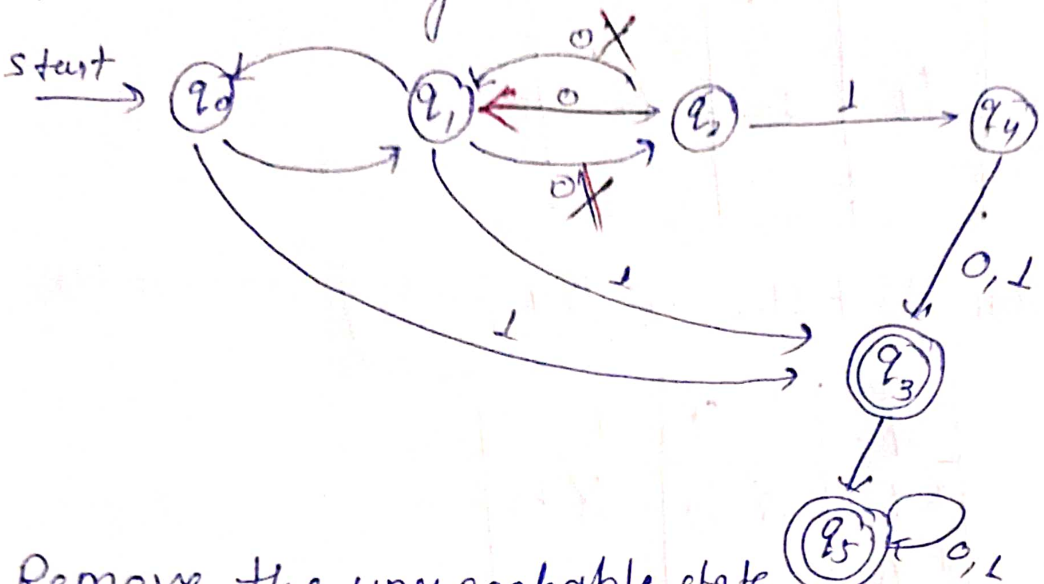
$\delta(q, a) = P$

⑤ Repeat step (IV) till there is no similar rows are available in T_1

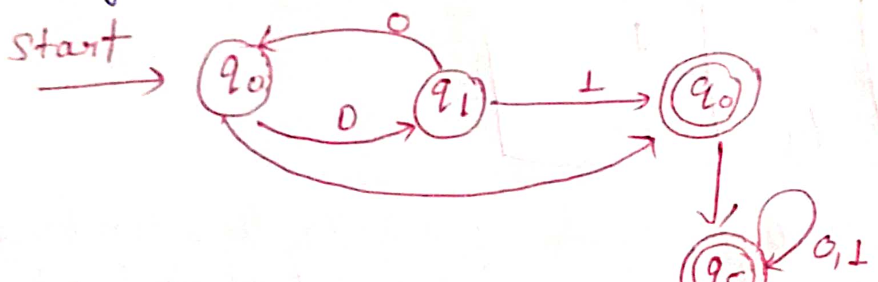
⑥ Repeat the ~~step~~ step ④ and step ⑤ for table T_2 also

⑦ Now combined the reduced T_1 and T_2. The combined transition table is transition table of minimized DFA

Q) Minimize the given DFA



① Remove the unreachable state q_2 and q_4 from the given DFA



② Draw the transition for rest of the states

$\delta \mid \Sigma$	a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3
$*q_3$	q_5	q_5
$*q_5$	q_5	q_5

③ split the transition table into two tables T_1 and T_2

T_1 =

$\delta \mid \Sigma$	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3

T_2 =

$\delta \mid \Sigma$	0	1
$*q_3$	q_5	q_5
$*q_5$	q_5	q_5

(4) Now consider the table T_1 for similar rows.

T_1 =

δ / Σ	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3

(5) Now consider the table T_2 for finding similar rows

T_2 =

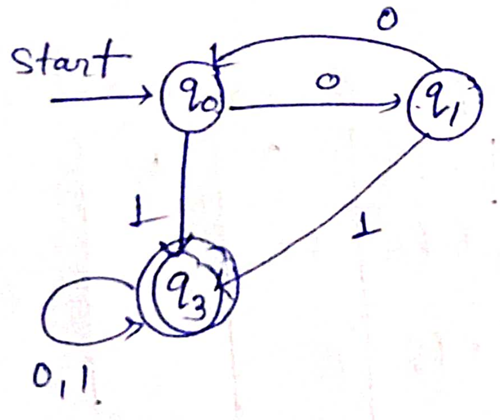
δ / Σ	0	1
$* q_3$	~~q_5~~ q_3	~~q_5~~ q_3
~~$* q_5$~~	~~q_5~~	~~q_5~~

T_2 =

δ / Σ	0	1
$* q_3$	q_3	q_3

(6) Now combine the reduced table T_1 and T_2 the combine transition table is the transition table of Minimized DFA

δ / Σ	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3
$* q_3$	q_3	q_3



Transition Diagram for Minimized DFA.

Finite state Machine (FSM) or Transducer :

A finite state M/C is similar to FA except that it has the additional capability of producing o/p.

FSM = FA + output capability

Types of Finite state Machine:

1. Moore M/C
2. Mealy M/C

- **Moore Machine**

If the output of FSM is dependent on percent state only then this Model of FSM is known as Moore M/C.

- **Mealy Machine**

If the output of FSM is dependent on present state and present input then this Model of FSM is known as Mealy M/C.

- **Moore M/C defination:**

A Moore M/c can be described by six tuple

$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

Q = Q is a finite set of states.

Σ = Σ is the input alphabet.

Δ = Δ is the output alphabet.

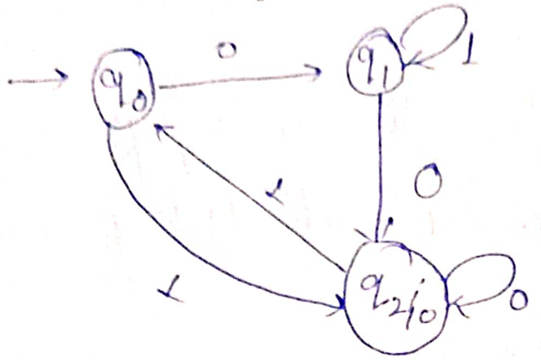
q_0 = q_0 is the initial state

δ = δ is transition function which maps $Q \times \Sigma \Rightarrow Q$

λ = λ is the output function mapping

$Q \rightarrow \Delta$

Q) Consider the Moore M/C shown in fig below. Construct the transition table.
What is the output for input string 0110?



Sol Transition Table

Present state P(S)	Input 0 Next state	Input 1 Next state	output λ
$\rightarrow q_0$	q_1	q_2	1
q_1	q_2	q_1	1
q_2	q_2	q_0	0

Now the transition sequence for the string 0110 is

$$\rightarrow q_0 \xrightarrow[1]{0} q_1 \xrightarrow[1]{1} q_1 \xrightarrow[1]{1} q_1 \xrightarrow[1]{0} q_2 \xrightarrow[0]{X}$$

So the o/p is 11110 for the input & the output is $\lambda(q_0) = 1$

Note :- For Moore M/C if the input string is of length 'n' the output string is of n+1.

Mealy Machine Defination

A mealy M/C can be described by six tuple

$(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

Q is finite and non empty set of states

Σ is input alphabet

Δ is output alphabet

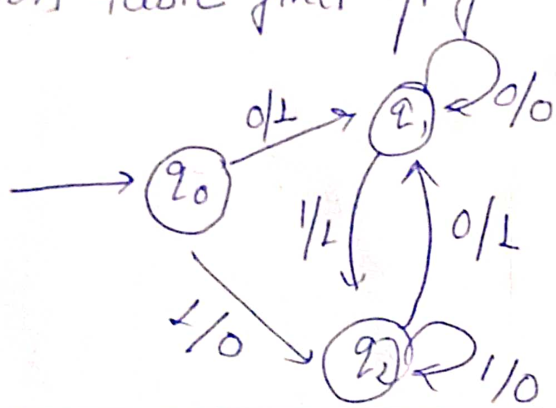
δ is transition function which maps $Q \times \Sigma \rightarrow Q$

λ is output function which maps $Q \times \Sigma \rightarrow \Delta$

q_0 is the initial state

Q) Consider the Mealy M/c shown is fig. Construct the transition table find o/p for input string 01010?

Sol



PS	Inputs 0 NS	0 o/p	1 NS	1 o/p
$\rightarrow q_0$	q_1	1	q_2	0
q_1	q_1	0	q_2	1
q_2	q_1	1	q_2	0

Now the transition sequence for input string 01010

$$\rightarrow q_0 \xrightarrow[1]{0} q_1 \xrightarrow[1]{1} q_2 \xrightarrow[1]{0} q_1 \xrightarrow[1]{1} q_2 \xrightarrow[1]{0} q_1$$

o/p $\rightarrow$ 11111

Q) Find the output for input string 0011?

PS	Input 0		Input 1	
	NS	O/P	NS	O/P
$\rightarrow q_1$	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

$$\rightarrow q_1 \xrightarrow[0]{0} q_3 \xrightarrow[1]{0} q_2 \xrightarrow[0]{1} q_4 \xrightarrow[0]{1} q_3$$

O/P → 0100

Equivalance of Moore and Mealy M/c:

The meaning of word equivalance is the two M/c which accepts the same lang. hence equivalance of Moore & Mealy M/c means both the M/c generate same o/p o/p string as same I/P string

Procedure for transforming a Moore M/c to corresponding to Mealy M/c.

Construction

① We have to define o/p function λ' for Mealy M/c as a function of present state.

we define $\lambda'(q,a) = \lambda(\delta(q,a))$ for all states q and input symbol a.

② The transition function is the same as that of given Moore M/c.

Q) Construct a Mealy M/C which is equivalent to the Moore M/C given in transition table.

Sol

Present State (PS)	Input 0 NS	Input 1 NS	O/P
→ q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

Present State (PS)	Inputs 0 NS	0 O/P	1 NS	1 O/P
→ q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0

① $\lambda'(q_0, 0) = \lambda(\delta(q_0, 0))$
$= \lambda(q_3) = 0$

② $\lambda'(q_0, 1) = \lambda(\delta(q_0, 1))$
$= \lambda(q_1) = 1$

Difference betwen Moore and Mealy M/C

[do it yourself]