UNIT - 1

Introduction to Finite Automata

Theory of Computation:-

It is a branch of Computer Science that deals with how efficiently a problem can be solved on a model of computation, using an algorithm.

Theory of Computation is mainly divided into 3 types.

1. Automata theory & language
2. Computability Theory
3. Complexity theory.

① **Automata theory and language**

It deals with the defination and Properties of various mathematical model of compitation.

Ex:- Finite Automata, Context free grammer, Turing Machine

② **Computability theory:-**

It deals with what can and can't be computed by the model.

③ **Complexity theory:**

It group the computable problem based on their hardness

Main Purpose of TOC:-

It develop formal mathematical model of computation that reflects real world Computers.

Scanned with CamScanner

Basic Definations

① Symbol:- It is a character

Ex:- a, b, c, z ..... z
0, 1, 2, ... 9
+, −, %, ... special characters.

② Alphabet:-

An alphabet is a finite, non-empty set of symbol. It is denoted by Σ (sigma)

Ex:- $\Sigma = \{0, 1\}$ set of binary alphabets.

$\Sigma = \{a, b, c, \ldots z\}$ set of all lowercase letters.

$\Sigma = \{+, *, \%, \ldots\}$ set of all special characters.

③ String (or) word:-

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

Ex:- a) 01110010 is a string from binary alphabet $\Sigma = \{0, 1\}$

b) aabbaacab is a string from alphabet $\Sigma = \{a, b, c\}$

Symbol	Alphabet	String (or) word
a, 1, 0	{0, 1}	01101001
	{a, b, c ...}	abbca

④ Empty string:-

The empty string is a string with zero occurance of symbol (no symbol). It is denoted by "ϵ" → no symbol string is { } empty.

Scanned with CamScanner

⑤ Length of String :-

It is the no. of symbols in the string. It is denoted by $|w|$

Ex :- $w = 0101101101$ from binary alphabet $\Sigma = \{0,1\}$

length of string $|w| = 9$

⑥ Concatenation of String :-

Join 2 or more strings side by side we call it as. concatenation of a string.

let $x = a_1 a_2 a_3 \ldots a_n$ &

$y = b_1 b_2 b_3 \ldots b_n$

Concatenation of string $xy = a_1 a_2 a_3 \ldots a_n b_1 b_2 b_3 \ldots b_n$

Ex :- $S = ababa$ & $t = edcddc$

Concatenation $st = ababacdcddc$

⑦ Power of an alphabet :-

If "Σ" is an alphabet, we can express set of all strings of certain length from that alphabet by using exponential notation.

It is denoted by Σ^k is the set of strings of length k.

Ex :- $\Sigma = \{0,1\}$ has 2 symbols

(i) $\Sigma^1 = \{0,1\}$ $(\because 2^1 = 2)$ $k=1$

ii) $\Sigma^2 = \{00, 01, 10, 11\}$ $(\because 2^2 = 4)$ $k=2$

iii) $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

$\therefore (2^3 = 8)$ $k = 3$

Scanned with CamScanner

The set of strings over an alphabet Σ is usually denoted by Σ^* = Kleen closure

for instance $\Sigma^* = \{0,1\}^*$

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

Kleen closure ($\because \Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$) with ϵ symbol

The set of strings over an alphabet Σ excluding ϵ is usually denoted by Σ^+ = Kleen plus

for instance $\Sigma^+ = \{0,1\}^+$

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

$(\because \Sigma^+ = \Sigma^* - \{\epsilon\})$

(OR)

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots\}$ without ϵ symbol.

Power of alphabet — Kleen closure (Σ^*)

Kleen plus (Σ^+)

(8) Languages:-

It is a finite set of non-empty string.

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

Ex:- The set of legal English words

The set of legal 'C' Programs

The set of string consisting of n 0's followed by n 1's...

$= \{\epsilon, 01, 0011, 000111, \ldots\}$

Scanned with CamScanner

Operations on Languages

① **Complementation:-**

Let L be a language over an alphabet Σ.

The complementation of L, denoted by $\bar{L}$

$$\boxed{\bar{L} = \Sigma^* - L}$$

② **Union:-**

Let L_1 and L_2 are languages over an alphabet Σ.

The union of L_1 & L_2 denoted by $L_1 \cup L_2$ is x/x is L_1 or L_2

③ **Intersection:-**

Let L_1 and L_2 are languages over an alphabet Σ

The intersection of L_1 and L_2 denoted by $L_1 \cap L_2$ is

$\{x/x$ in L_1 & $L_2\}$

④ **Concatenation:-**

Let L_1 and L_2 be languages over an alphabet Σ.

The Concatenation of L_1 & L_2 denoted by

$L_1 \cdot L_2$ is $\{w_1 \cdot w_2 / w_1$ is in L_1 & w_2 is in $L_2\}$

⑤ **Reversal**

Let L be a language over an alphabet Σ. The

Reversal of L denoted by

L^r, is $\{w^r / w$ is in $L\}$

Scanned with CamScanner

⑥ Kleen's closure

Let 'L' be a language over an alphabet Σ.

The Kleen closure of L, denoted by

$L^* = \{ x / $ for an integer $n \geq 0 \}$

$x = x_1 x_2 \ldots x_n$ & $x_1 x_2 \ldots x_n$ are in L

$L^* = \bigcup_{i=0}^{\infty} L^i$ (Eg:- $a^* = \{ \epsilon, a, aa, aaa, \ldots \}$)

⑦ Positive closure :-

Let L be a language over an alphabet Σ.

The closure of L, denoted by L^+ is

$\{ x /$ for an integer $n \geq 1$, $x = x_1, x_2 \ldots x_n$ & $x_1, x_2 \ldots x_n$ are in L)

$L^+ = \bigcup_{i=1}^{\infty} L^i$ (Eg:- $a^+ = \{ a, aa, aaa, \ldots \}$)

Scanned with CamScanner

Finite Automata (FA)

Finite Automata is an abstract computing device. It is a mathematical model of system with discrete inputs, outputs, states and set of transitions from state to state that occurs on input symbol from alphabet Σ.

It representations :-

→ Graphical (Transition Diagrams (or) Transition table)

→ Tabular (Transition Table)

→ Mathematical (Transiton function or Mapping function)

Formal Defination of Finite Automata

A finite automata is a 5-tuples, they are

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

Where

Q = is a finite set called the states

Σ = is a finite set called the alphabets

$\delta = Q \times \Sigma \to Q$ is the transition function.

$q_0 \in Q$ is the start state also called initial state

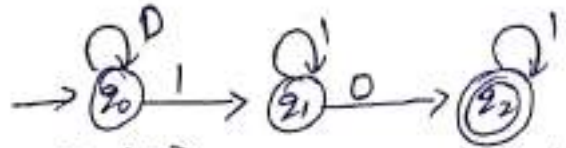
$F \subseteq Q$ is the set of accept states, also called final state.

Scanned with CamScanner

Transition Diagrams (Transition Graph)

It is a directed graph associated with the vertices of the graph corresponds to the state of finite automata.

Eg:-



Start (or) Initial state

Final (or) Accepting state

{0, 1} are inputs

q_0 — initial state.

q_1 — intermediate state

q_2 — Final state.

Transition Table

It is basically a tabular representation of the transition function that takes two arguments (a state and a symbol) and returns a value (the "next state")

Transition Table Consists of

→ Rows corresponds to states

→ column corresponds to input symbols

→ Entries corresponds to next states

→ The next state is marked with an arrow (→)

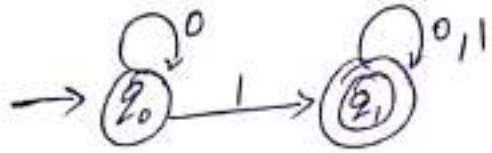
→ The accept state are marked with a star (*)

	Σ	Σ	Σ	Σ	Σ
→ Q	✓	✓	✓	✓	✓
Q	✓	✓	✓	✓	✓
Q	✓	✓	✓	✓	✓
Q	✓	✓	✓	✓	✓
* Q	✓	✓	✓	✓	✓

→ next states

$\delta = Q \times \Sigma \rightarrow Q$

Scanned with CamScanner



q_0 — initial

q_1 — Final

	0	1
$\rightarrow q_0$	q_0	q_1
$*q_0$	q_1	q_1

Transition Function

* The mapping function of transition function denoted by δ
* Two parameters are passed to this transition function
  i) Current state
  ii) Input Symbol
* The transition function returns a state which can be called as next state.

δ (Current-State, Current-Input Symbol) = next-state

$$Q \times \Sigma \rightarrow Q$$

Example:

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

(or)

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

Applications of Finite Automata

* It plays an important role in compiler design.
* In switching theory and design and analysis of digital circuits automata theory is applied.
* Design and analysis of Compiler s/w and h/w Systems.

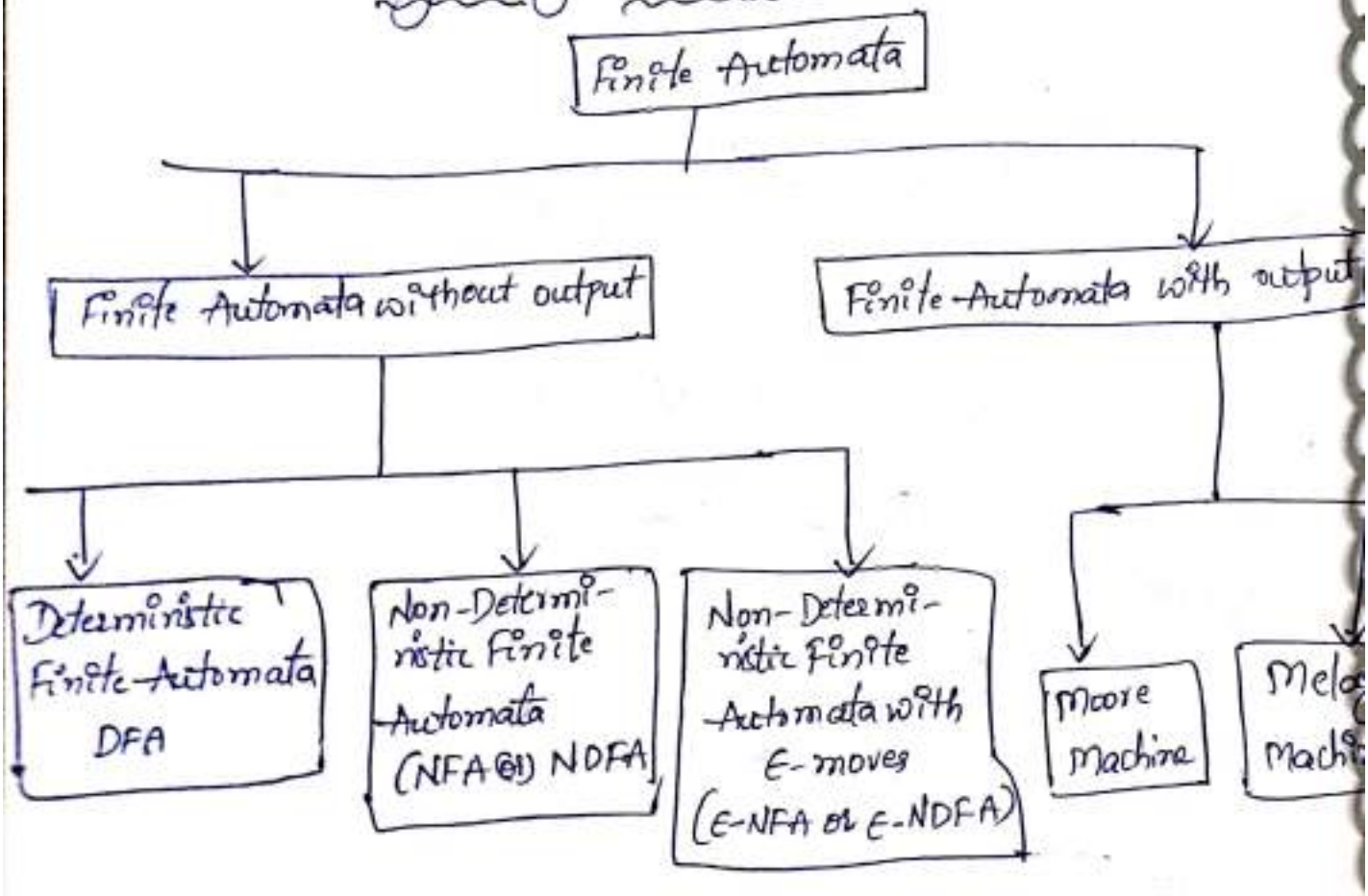
Scanned with CamScanner

* To Prove Correctness of the program automata theory is used.

* To design finite state machines such as Moore & Melay machine.

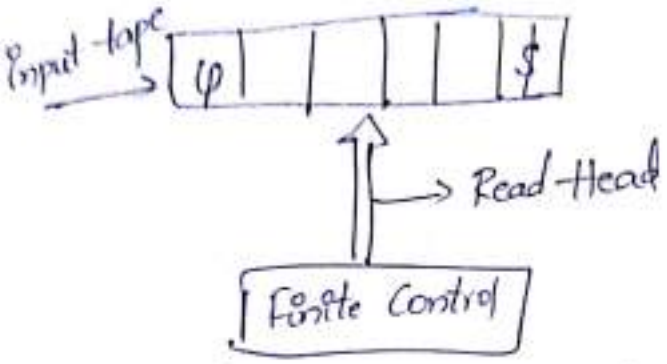
* It is base for the formal language and these language are useful of the programming languages.

Types of Finite Automata



Scanned with CamScanner

Finite Automata Model:-



It Contains 3 Components

1) Input tape
2) Reading Head (Read)
3) Finite Control (control unit)

Input tape:- Input tape divides into cells, each cell stores only one unit ie. one input symbol.

* each cell contains only one symbol
* Input buffer contains 2 special symbols used to represent end markers.

φ (symbol) = used to represent begining of the Input tape

$ – used to represent end of the Input tape.

(Reading head)

It is used to an Input symbol from the input tape.

* Initially Reading head points to first cell of the input read.

Scanned with CamScanner

Finite Control unit:

* It is used to manage entire finite automata.
* Initially finite automata is available at initial state and by reading some input the control moves to next state. the read head point to next input cell.
* Always reading head moves in a forward direction only.

⟹ In Finite automata states are used to describe the behaviour of the system, it can be classified into 4 types.

1. Initial state [only 1 initial state for machine]
2. Final state [more than one]
3. Intermediate
4. Hand/death [outgoing transition]

Finite automata representation

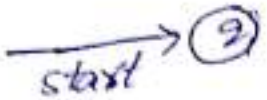
⟹ It can be represented in 2 ways

1. Transition diagram
2. Transition Table

⟹ The pictorial representation of finite automa is called Transition diagram.

Scanned with CamScanner

⇒ The initial state can be represented as



⇒ The final state is represented with double circle (*)

Ex :- 

⇒ Transition is in the form

$$\delta(q, a) = p$$

$\delta(q_0, b) = q_0$ ⇒ 

Transition Table

The tabular representation of finite automata states can be represented as rows, input alphabets is act a columns.

Ex: Construct transition diagram & Table for the following Machine.

$$M = (\{q_1, q_2, q_3, q_4\}, \{0,1\}, \delta, q_1, q_4)$$

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

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

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

$\delta(q_4, 0) = q_3$

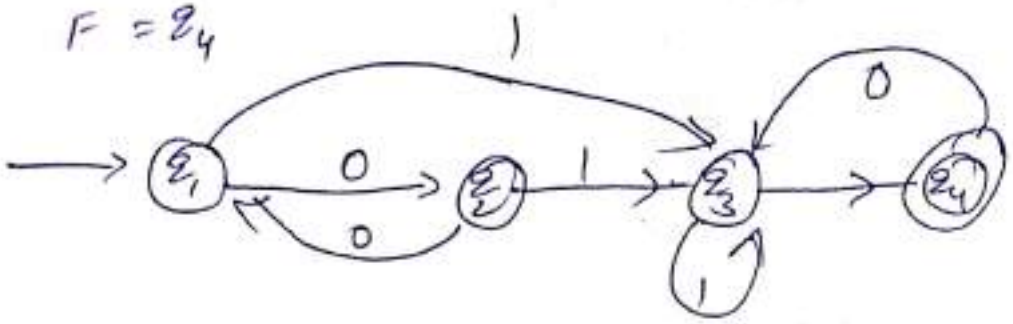
Scanned with CamScanner

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

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

$q_0 = q_1$

$F = q_4$



	0	1
$\rightarrow q_1$	q_2	q_3
q_2	q_1	q_3
q_3	q_4	q_3
$*q_4$	q_3	—

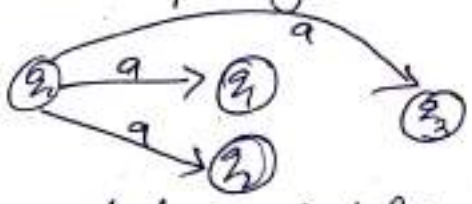
Scanned with CamScanner

Types of Finite Automata

Based on transition Finite automata (FA) are classified into 2 types.

1. Deterministic FA (DFA)
2. Non-Deterministic (NFA)

Difference b/w DFA & NFA

DFA	NFA
→ A DFA can be defined as 5 tuples $M: (Q, \Sigma, \delta, q_0, F)$ $\delta = Q \times \Sigma \longrightarrow Q$	→ A NFA can be defined as 5 tuples $M = (Q, \Sigma, \delta, q_0, F)$ $\delta : Q \times \Sigma \longrightarrow 2^Q$
→ In case of DFA, every transition generates single outcome with a particular input symbol 	→ In case of NFA, transition generates more than one outcome with same input symbol. 
→ In case of DFA, every state must contains one (a) transition with every input symbol in the given alphabet.	→ No need to maintain transitions with every input symbol.
→ It is difficult to construct	→ It is easy to construct.
→ Epsilon transition are not allowed.	→ E-transition are allowed.

Scanned with CamScanner

Deterministic Finite Automata (DFA)

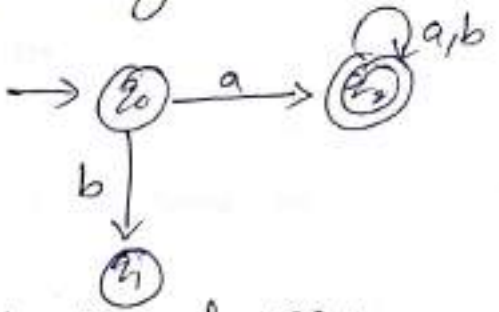
→ The Finite Automata are called deterministic finite automata if the machine is read an input string one symbol at a time.

→ Deterministic refers to the uniqueness of the computation.

→ In DFA, there is only one path for specific input from the current state to the next state.

→ DFA does not accept the null move, i.e. DFA can't change state without any input character.

→ DFA can contain multiple final states. It is used in lexical analysis in compiler.



Formal Defination of DFA :-

A DFA is a collection of 5 tuples. (Same as FA)

Q : Finite set of states

Σ : Finite set of input symbols.

q_0 : Initial state

F : Final state

δ : Transition function.

Transition function can be defined as $\delta : Q \times \Sigma \to Q$

Scanned with CamScanner

Graphical Representation of DFA

DFA can be represented by diagraphs called state diagram.

1. The state is represented by vertices.
2. The arc labeled with an input character show the transition.
3. The initial state is marked with an arrow.
4. The final state is denoted by a double circle.

Acceptance of Language:

* A Language acceptance is defined by "if a string w is accepted by the machine M. i.e. if it is reaching the final state F by taking the string w.
* Not accepted if not reaching the final state.

Ex:- $L = \{\phi\} \Rightarrow$ → (q0) (no string)

$L = \{\epsilon\} \Rightarrow$ → (q0)

DFA to accept "a" ⇒ → (q0) —a→ (q1)

{ states depends on length of string }

no. of state = [min string length + 1]

DFA to accept zero or more "a"

$L = \{\epsilon, a, aa, aaa, \ldots\}$

→ (q0) with a self-loop a

Scanned with CamScanner

① Let M be the given finite automata & 'w' is the given string then to check string acceptance use the following

$$\delta(q_0, w) \Rightarrow q_r \quad , q_r \in F \text{ (final state)}$$

for Consider $w = 101$, check whether string is accepted or not.

$$\delta(q_0, 101) \vdash \delta(q_1, 01)$$
$$\vdash \delta(q_2, 1)$$
$$\vdash (q_3) \in F$$

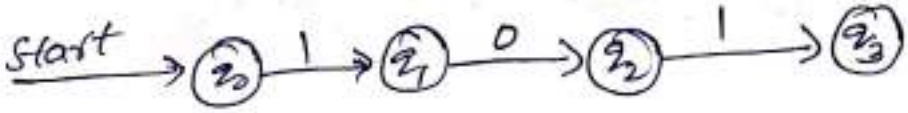
∴ String is accepted

$w = 1011$

$$\delta(q_0, 1011) \vdash \delta(q_1, 011)$$
$$\vdash \delta(q_2, 11)$$
$$\vdash \delta(q_3, 1)$$

Dhang

∴ String is rejected.



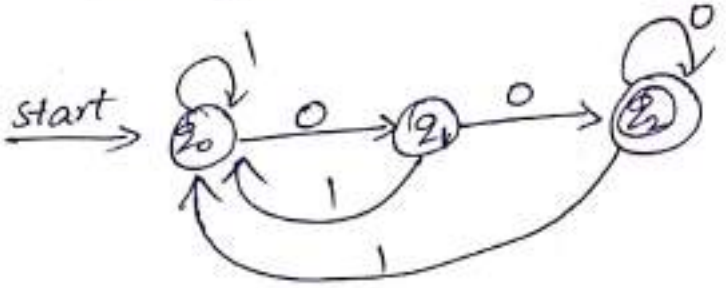
Scanned with CamScanner

② Design DFA to accept all the strings ends with {00} over the alphabet $\Sigma = \{0, 1\}$

Sol: Given alphabet $\Sigma = \{0, 1\}$

$L = \{00, 000, 100, 0100, 0000, 1000, 1100, \ldots\ldots\}$

we require three states



Transition table

	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

$\delta(q_0, 000101) \vdash \delta(q_1, 00101)$

$\vdash \delta(q_2, 0101)$

$\vdash \delta(q_2, 101)$

$\vdash \delta(q_0, 01)$

$\vdash \delta(q_1, 1)$

$q_0 \notin$

$\therefore$ String is rejected

Scanned with CamScanner

$w = 01001000$

$\delta(q_0, 01001000) \vdash \delta(q_1, 1001000)$

$\delta(q_0, 001000)$

$\delta(q_1, 01000)$

$\delta(q_2, 1000)$

$\delta(q_0, 000)$

$\delta(q_1, 00)$

$\delta(q_2, 0)$

$q_2 \in F$.

$\therefore$ String is accepted.

③ Draw DFA for the following

$Q = \{a, b, c\}$, $\Sigma = \{0, 1\}$ $q_0 = \{a\}$, $F = \{c\}$

Present state	0	1
→ q	a	b
b	c	q
*c	b	c

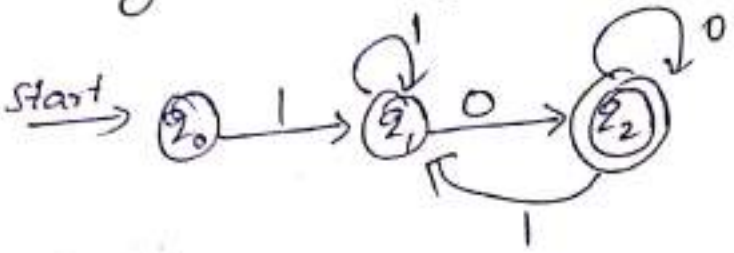
Sol :-

Scanned with CamScanner

③ Design DFA with $\Sigma = \{0, 1\}$ accept those string which start with 1 and end with 0.

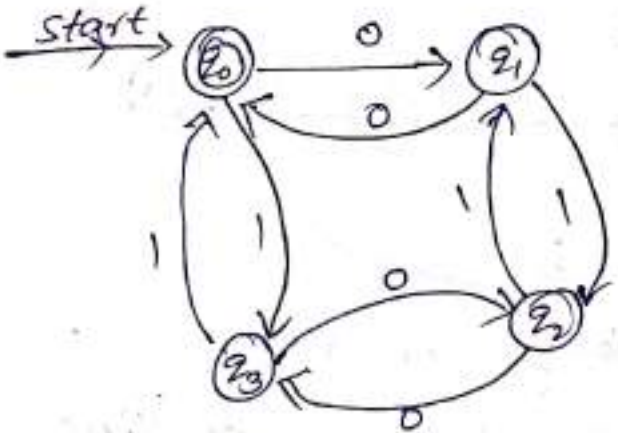
Sol:- $L = \{10, 100, 1010, \ldots\}$

Min. length = 2, no. of states = 2+1 = 3



④ Design DFA with $\Sigma = \{0, 1\}$ accepts even no. of 0's and even no. of 1's.

Sol:- $L = \{0011, 1100, 1010, 1001, \ldots\}$



q_0: State of even no. of 0's & even no. of 1's. → accepted

q_1: State of odd no. of 0's & even no of 1's

q_2: State of odd no. of 0's & odd no. of 1's

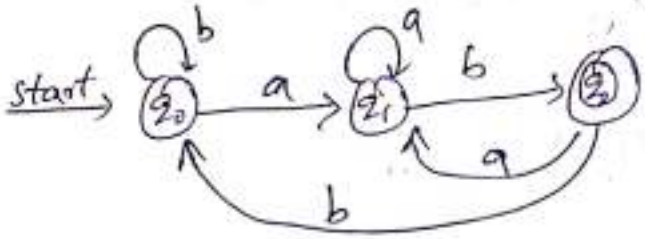
q_3: State of even no. of 0's & odd no of 1's

Scanned with CamScanner

⑤ Construct DFA accepting all strings over {a, b}, the string is ending with 'ab'?

Sol: $L = \{ab, aab, bab, \ldots\}$

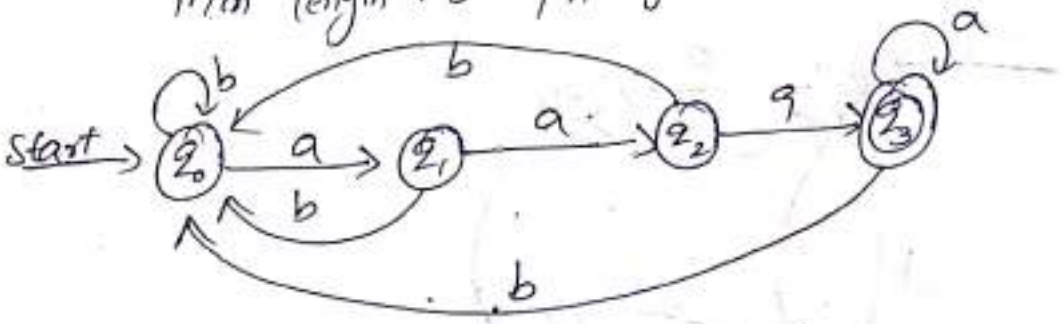
min. length of string = 2, no. of states = 2+1 = 3



⑥ Give DFA which reads strings from {a, b} and ends with aaa.

Sol: $L = \{aaa, baaa, bbaaa, babaaa, \ldots\}$

min length = 3, no. of states = 3+1 = 4



⑦ Design DFA for the following over {0, 1}

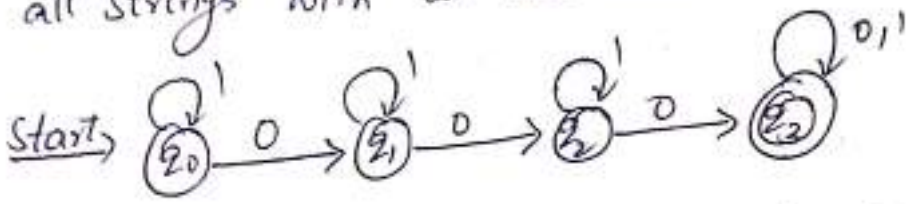
a) All strings containing not more than three 0's.

b) All strings that has atleast two occurrences of 1 between any two occurrences of 0.

Scanned with CamScanner

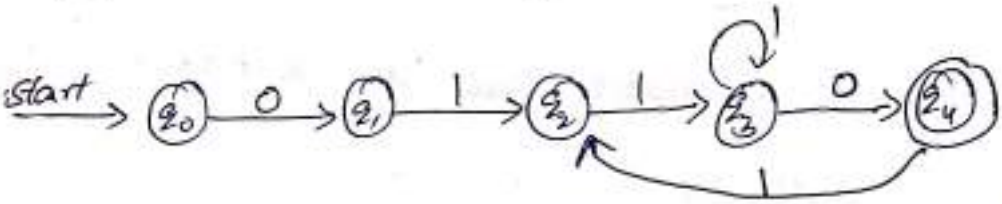
Sol:-

a) Let us first design DFA for the language containing all strings with at the most three 0's.



Now, for the DFA which accepts all strings containing not more than three 0's we will convert non-final state to final state and final state to non-final state.

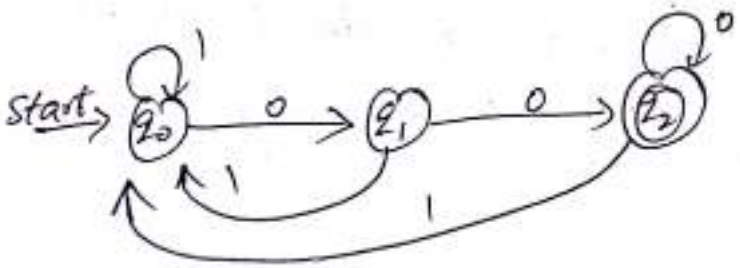
b) The DFA with at least two occurrences of 1 between any two occurrences of 0 is



(8) Design DFA to accept all the strings ends with {00} over the alphabet $\Sigma = \{0,1\}$

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

$L = \{00, 000, 100, 1000, 0100, 0000, 1100, \ldots\}$

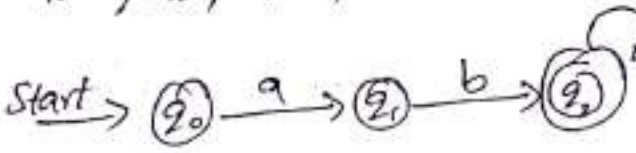


Scanned with CamScanner

(9) Design DFA to accept strings over an alphabet $\{a, b\}$ and every strings start with 'ab'.

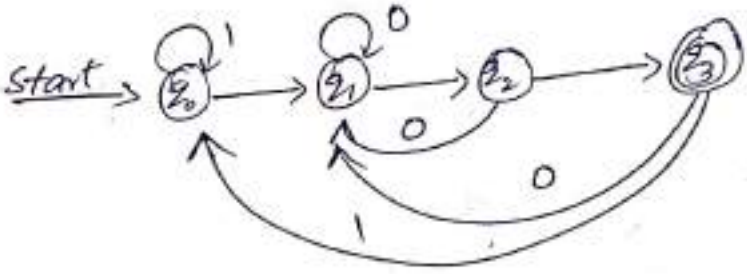
Sol:- Given alphabet $\Sigma = \{a, b\}$

$L = \{ab, aba, abb, abab, abaa, abbb, \ldots\}$



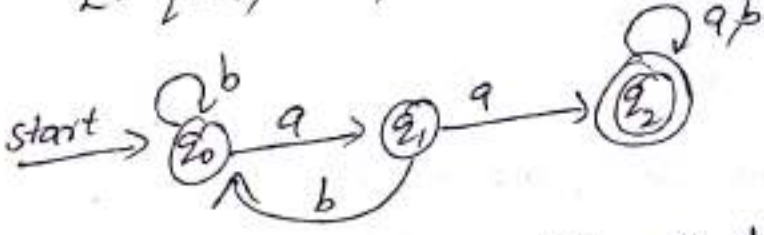
(10) Design DFA to accept all strings over an alphabet $\{0, 1\}$ and every string ends with 011

Sol:- $L = \{011, 0011, 1011, 11011, 10011, \ldots\}$



(11) Design a finite automata over an alphabet $\{a, b\}$ and every string having substring aa

Sol:- $L = \{aa, aab, baa, aaba, aaab, baab, \ldots\}$



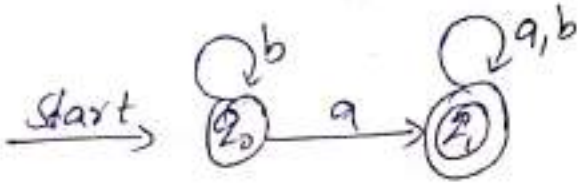
(12) Design a DFA to contain all the strings over an alphabet $\{a, b\}$ and having substring abb.

Scanned with CamScanner

Sol:- $L = \{abb, aabb, babb, ababb, \ldots\}$

(13) Design a DFA to accept atleast one 'a' over an alphabet $\{a, b\}$

Sol:-



(14) Design DFA to accept exactly one 'a' over an $\{a, b\}$

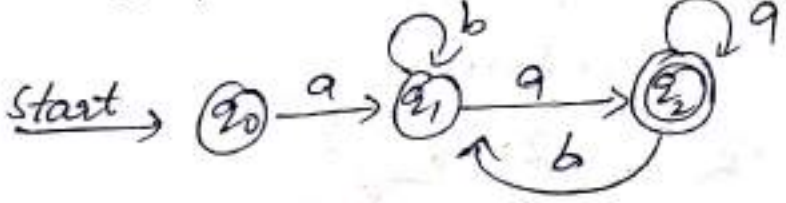
(15) Design a DFA to accept the following

$$L = \{awa, / w \in \{a, b\}^*\}$$

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

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

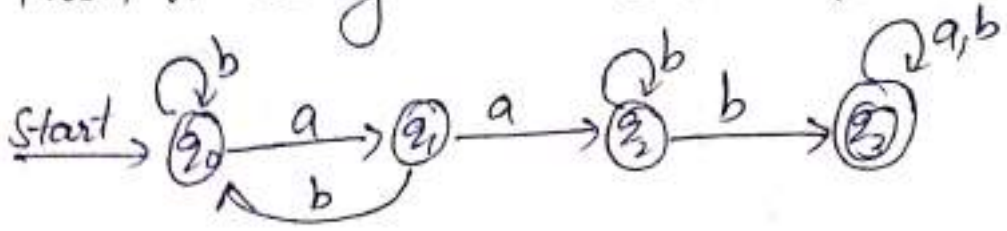
$L = \{aa, aaa, aba, aaba, abaa, \ldots\}$



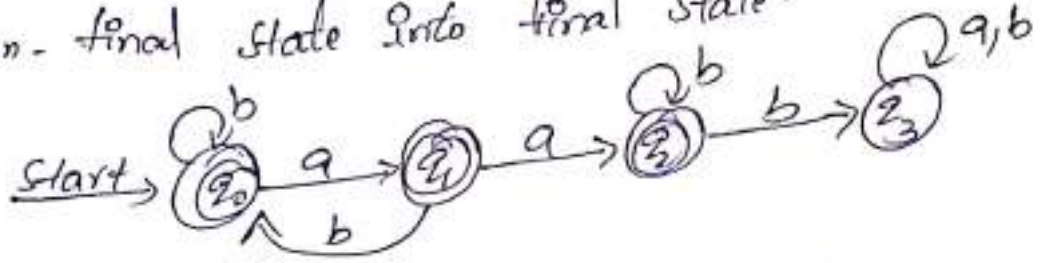
(16) Design DFA which accepts all strings over an alphabet $\{a, b\}$ except those containing 'aab'.

Scanned with CamScanner

Sol:- First, the strings which contain aab



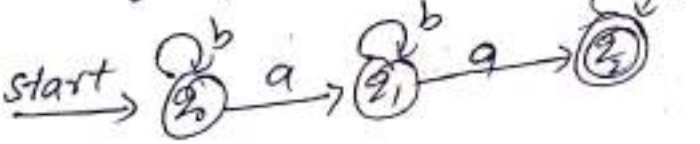
Now, for the strings which except those containing the 'aab', convert final state to non-final state and non-final state into final state.



(17) Design DFA which accepts following language

$L = \{w/w$ contains exactly two 'a's$\}$

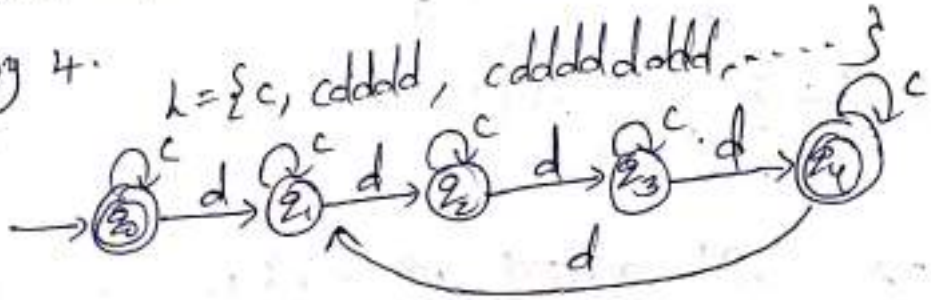
Sol: $L = \{aa, aab, aabb, \ldots\}$



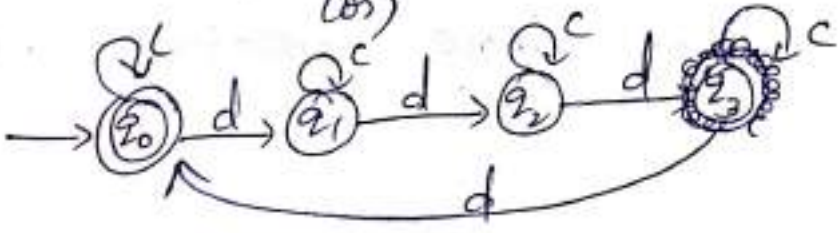
(18) Design DFA to accept a string over an alphabet $\{c, d\}$ such that the no. of d's in the every string divisible by 4.

$L = \{c, cdddd, cddddddddd, \ldots\}$

Sol:



(OR)



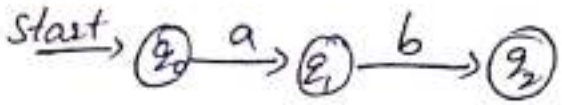
Scanned with CamScanner

(19) Design DFA to accept the following

$L = (ab)^n$, $n \geq 0$

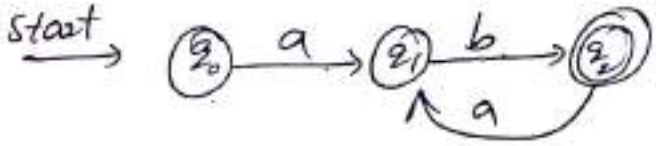
$L = (ab)^n$, $n \geq 1$

Sol:- $L = (ab)^n$, $n \geq 0$ i.e. $L = \{\epsilon, ab, abab, ababdb, \ldots\}$



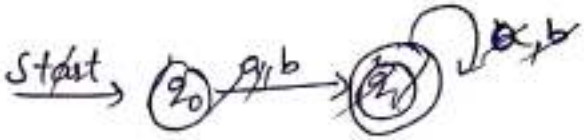
i.e.

$L = (ab)^n$, $n \geq 1$, i.e. $L = \{ab, abab, ababab, \ldots\}$



(20) Design $L = a^n ub$, $n \geq 1$

Sol:- $L = \{ab, aab, aaab, \ldots\}$



(21) Design DFA to accept odd no. of 1's and any no. of 0's in $\Sigma = \{0, 1\}$

Scanned with CamScanner

Sol:

$L = \{1, 10, 01, 1110, 0111, 00111, \ldots\}$

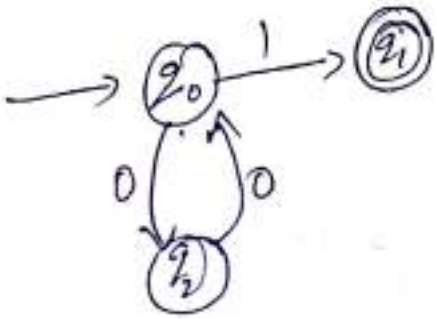
(22) Design DFA to accept the following $L = \{0^m 1^n \mid m \geq 0, n \geq 1\}$

Sol: $L = \{1, 011, 00111, 0001111, \ldots\}$

(23) Design DFA to accept even no. of 0's followed by single 1.

Sol: $L = \{1, 001, 0001, 000001, \ldots\}$

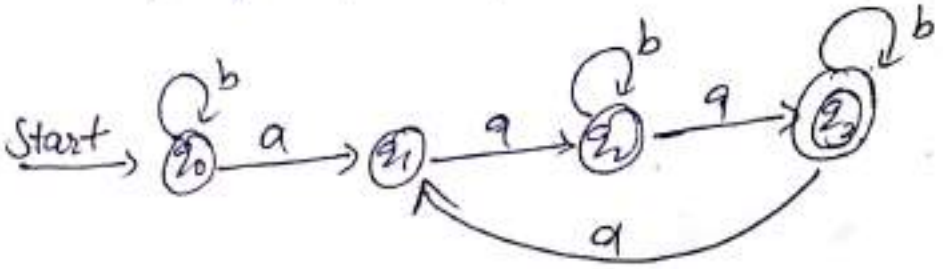
(OR)



Scanned with CamScanner

(24) Design DFA to accept L = { strings in which 'a' always appears triple.

Sol:- $L = \{aaa, baaa, aaab, aaaaaab, baaaaa a, \cdots\}$

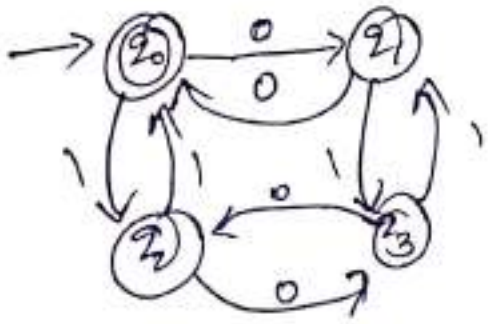


(25) Design DFA to check whether the given no. in binary number is even or not.

Sol:- If the binary number is ends with "0" then it is even number.

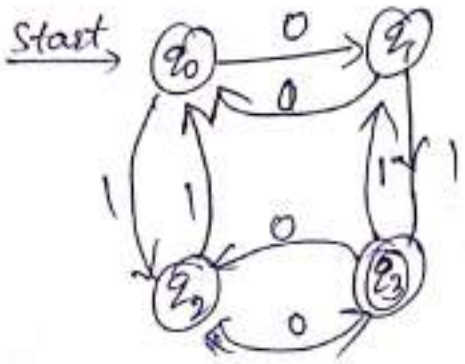
(26) Design DFA to accept even no. of 0's and 1's over an alphabet {0,1}

Sol:- $L = \{\epsilon, 0011, 1100, 1010, 0101, \cdots\}$



Scanned with CamScanner

(27) Design DFA it contains odd no. of 1's & 0's when q_3 is final state.



q_0 : even no. of 1's & 0's

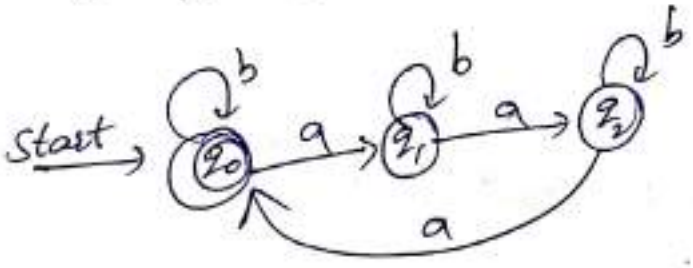
q_1 : odd no. of 0's & even no. of 1's

q_2 : even no. of 0's & odd no. of 1's

q_3 : odd no. of 0's & 1's.

(28) Design FA to accept string with 'a' and 'b' such that the number of a's are divisible by 3.

Sol:- $L = \{aaa, baaa, baaaaaa, aaabaaa, \ldots\}$



Scanned with CamScanner

(29) Design DFA for the following languages $\Sigma = \{a, b\}$

a) $L = \{w/w$ docsnot contains the substring $ab\}$

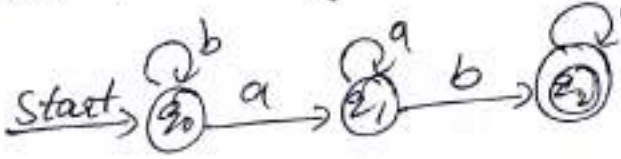
b) $L = \{w/w$ Contains neither the substring ab nor $ba\}$

c) $L = \{w/w$ is any string that docsn't contain exactly two a's$\}$

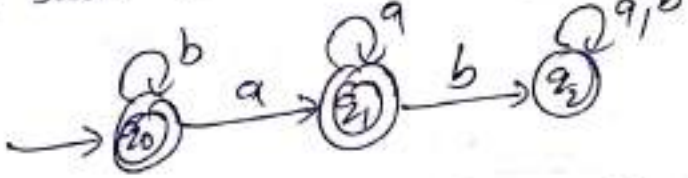
d) $L = w/w$ is any string except a and $b\}$

Sol:-

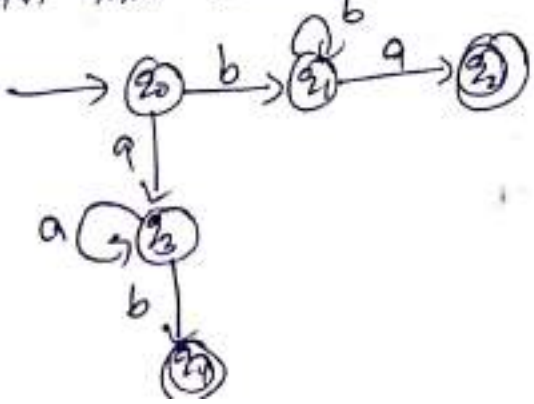
a) For designing the DFA specified by language L we will first design DFA that contains substring ab



Now, for DFA that accepts a language L which do not Contain Substring ab, we will simply change the accept State (final State) into a non-final state and non-final states will be made final states.

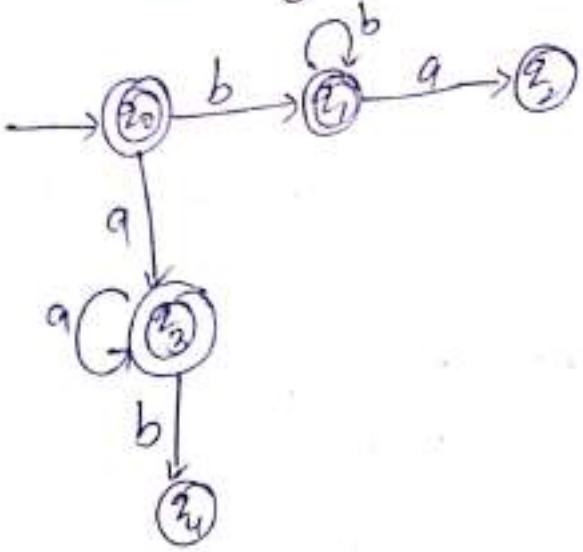


b) The DFA that Contains Substring ab or ba is

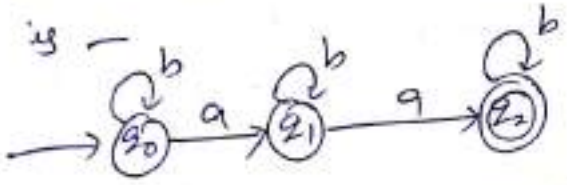


Scanned with CamScanner

Now if we change non-final state to final state and final state into non-final state then the DFA will be as follows. This DFA accepts a language which do not contain the substring ab or ba.

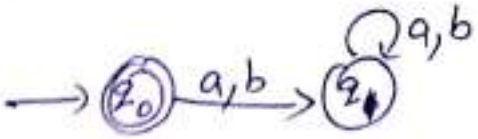


c) The DFA that contains a language having two a's exactly is –



Now, we will exchange the final and non-final states and then the DFA which will be obtained will represent a language that doesn't contain exactly two a's.

d) The DFA that doesn't contain a and b.



Scanned with CamScanner

(30) Design a DFA to accept strings with 0's and 1's such that the number of 1's should be even.

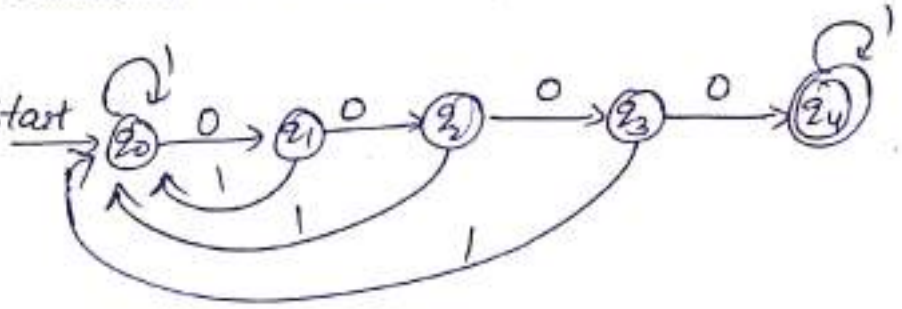
Sol:-

(31) Construct DFA accepting the set of all strings ending with 00.

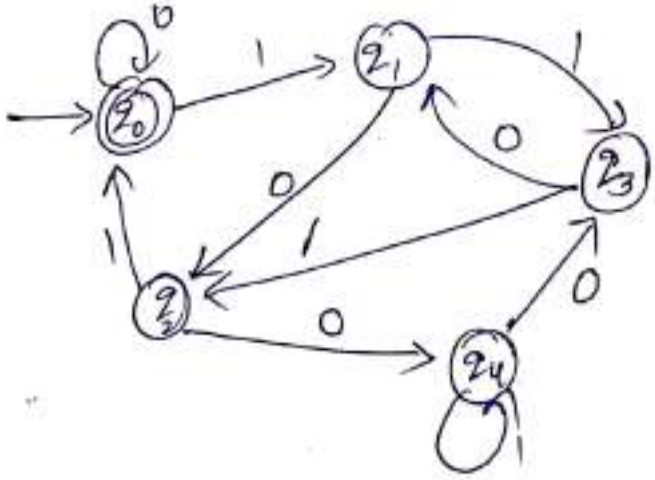
Sol:-

(32) Design FA to accept string over $\{0,1\}$ containing 4 consecutive zero's

Sol:-



(33) Construct DFA for accepting binary number which is divisible by 5 over an alphabet $\{0,1\}$



Note that for accepting string 10100 i.e $\delta(q_0, 10100) \vdash \delta(q_1, 0100)$
$\vdash \delta(q_2, 100)$
$\vdash \delta(q_0, 00)$
$\vdash \delta(q_0, 0)$
$\vdash q_0$ Final
Accepted.

Scanned with CamScanner

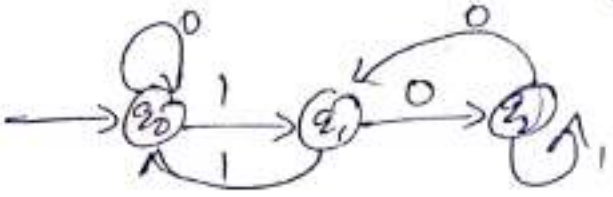
(34) Design DFA to accept all binary strings divisible by 3, $\Sigma = \{0,1\}$

Sol: The possible remainders are 0, 1, 2 i.e. so we require 3 states.

assume for remainder $0 \longrightarrow q_0$

$1 \longrightarrow q_1$

$2 \longrightarrow q_2$



$\rightarrow$ 0%3 = 0 (q_0) 10 => 2%3 = 2 100 => 4%3 = 1 (q_1)

=> 1%3 = 1 (q_1) 11 => 3%3 = 0 101 => 5%3 = 2 (q_2)

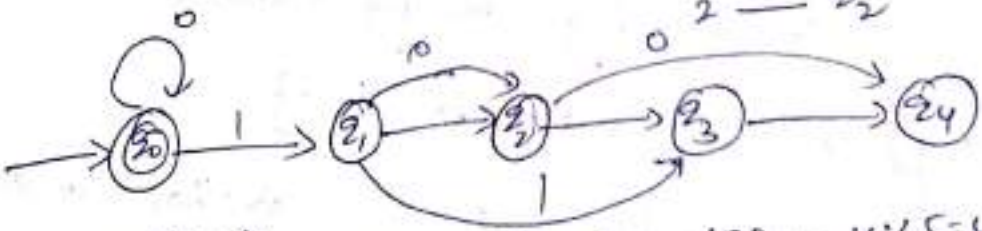
(35) Design DFA binary number not divisible by 5, $\Sigma = \{0,1\}$

Sol:- The possible states are 0, 1, 2, 3, 4, so we require 5 states.

$0 \longrightarrow q_0$ $3 \longrightarrow q_3$

$1 \longrightarrow q_1$ $4 \longrightarrow q_4$

$2 \longrightarrow q_2$

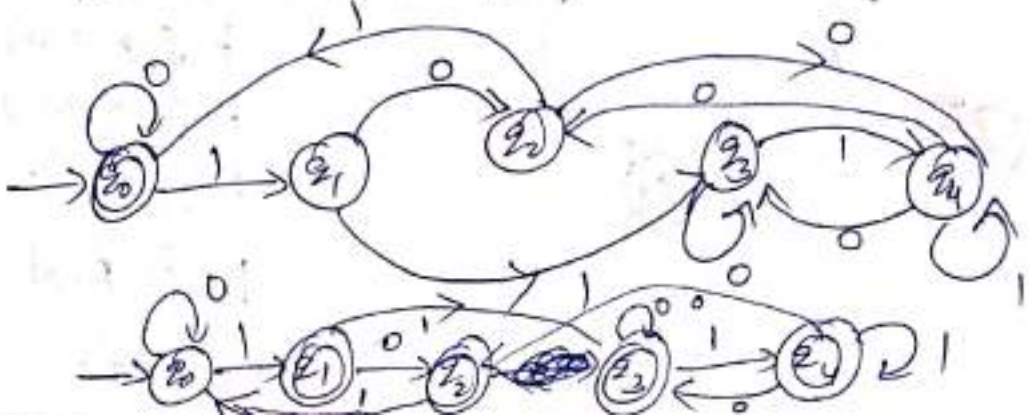


0 => 0%5 = 0 10 = 2%3 = 2 100 = 4%5 = 4 1000 = 8%5 = 3

1 => 1%5 = 1 11 = 3%5 = 3 101 = 5%5 = 0 1001 = 9%5 = 4

10010 = 18%5 = 3

10011 = 19%5 = 4



Scanned with CamScanner

NFA (Non-Deterministic Finite Automata)

→ The FA are called NFA, when there exist many paths for specific input from the current state to the next state.

→ It is easy to construct NFA than DFA for a given regular language.

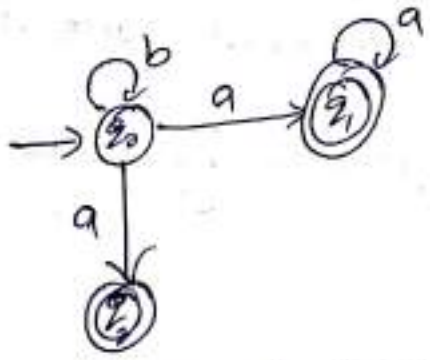
→ Every NFA is not DFA, but each NFA can be translated into DFA.

→ NFA is defined in the same way as DFA, but with two exceptions.

→ It contains multiple next state &

→ It contains ϵ-transitions.

Ex:-



Formal Defination of NFA:-

NFA also have five states same as DFA, but with different transition function.

$$\delta = Q \times \Sigma \longrightarrow 2^Q$$ → multiple states

Scanned with CamScanner

where,

Q : Finite set of states

Σ : Finite set of the input Symbol

q_0 : Initial State

F : Final State

δ : Transition Function.

Graphical Representation of NFA:-

An NFA can be represented by graphs called State diagram. In which

1. The state is represented by vertices.
2. The arc labeled with an input character show the transition.
3. The initial state is marked with an arrow.
4. The final state is denoted by double circle.

Scanned with CamScanner

Examples of NFA

(1) Design NFA transition table for the given NFA.

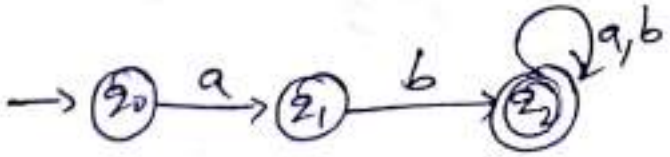
Sol:- $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, $q_0 = \{q_0\}$, $F = \{q_2\}$

State	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_1\}$
q_1	q_2	q_0
$*q_2$	q_2	$\{q_1, q_2\}$

(2) Construct NFA where $L = \{$ start with 'a' $\}$ and $\Sigma = \{a, b\}$

Sol:- $L = \{a, ab, abb, aba, abbb, abab, \ldots\}$

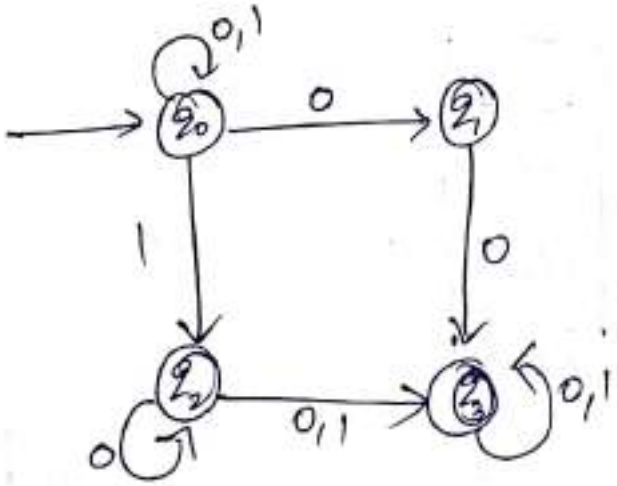
(3) Construct NFA, $L = \{$ start with ab $\}$, $\Sigma = \{a, b\}$



Scanned with CamScanner

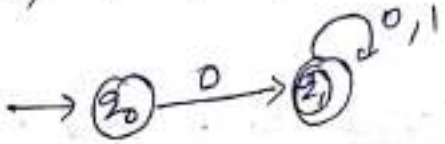
④ Design NFA for transition table

State	0	1
→ q_0	q_0, q_1	q_0, q_2
q_1	q_3	$\in$
q_2	q_2, q_3	q_3
* q_3	q_3	q_3



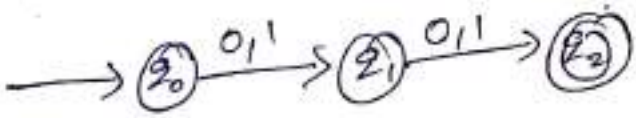
⑤ Construct NFA, $L = \{$ set of all strings that start with $\{0\}$ over $\Sigma = \{0,1\}$

Sol: $L = \{0, 00, 01, 001, 010, 011, 000, \ldots\}$



⑥ Construct NFA that accepts set of all strings over $\{0,1\}$ of length 2.

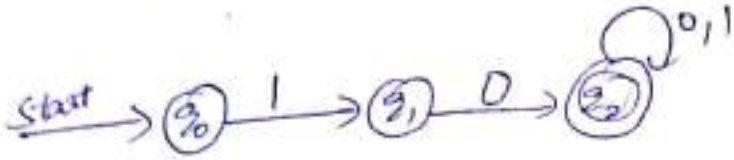
Sol:- $L = \{00, 01, 10, 11\}$, $\Sigma = \{0,1\}$



Scanned with CamScanner

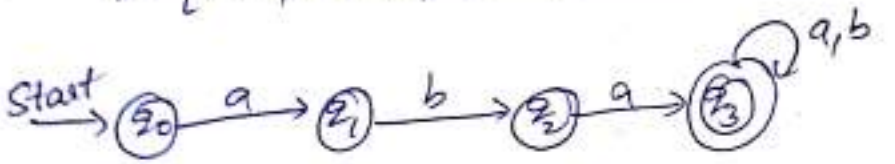
(7) Construct NFA that accepts set of all strings over {0,1} that starts with 10.

Sol:- $L = \{10, 100, 101, 1010, 1011, \ldots\}$



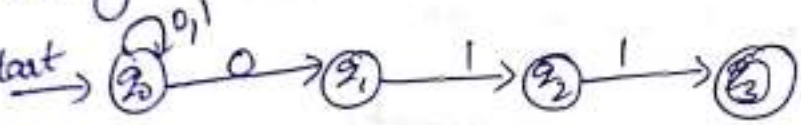
(8) Design NFA to accept all the strings over an $\Sigma = \{a, b\}$ and every string starts with aba.

Sol:- $L = \{aba, abaa, abab, ababa \ldots\}$



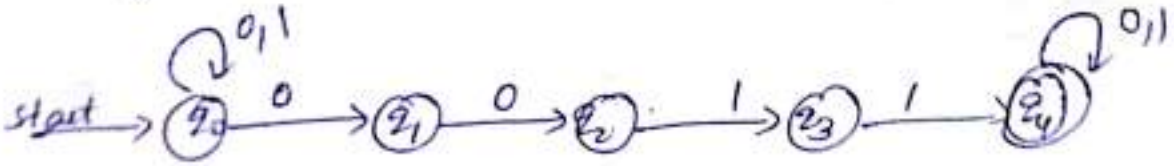
(9) Design NFA to accept all strings over {0,1} and every string ends with 011

Sol:-



(10) Design NFA to accept all the strings containing a substring 0011

Sol:-



Scanned with CamScanner

(11) Design NFA for the given language

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

Sol: $L = \{0101^n \cup 0100\}$

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

$L = \{010, 0100, 0101, 01011, \ldots\}$

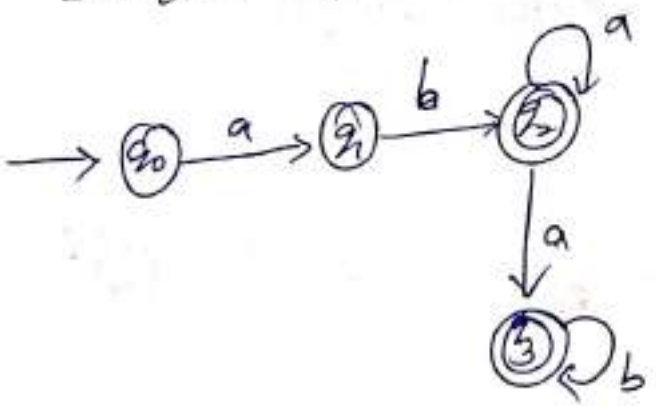
(12) Design NFA to accept language

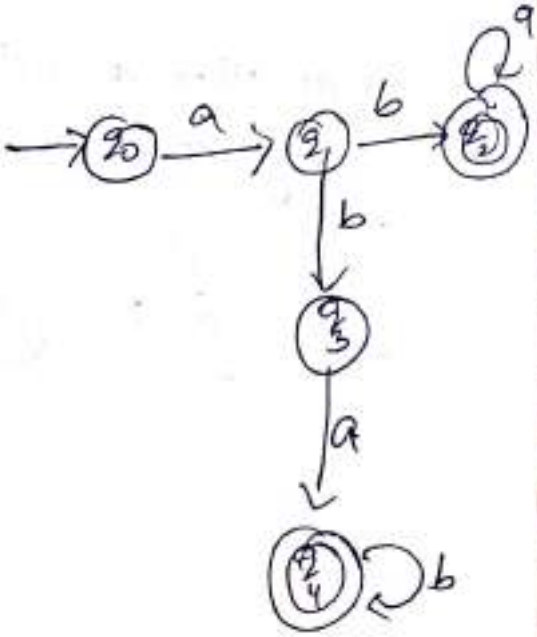
$L = \{w / w \in abab^n \text{ or } aba^n\}, n \geq 0$

Sol: $L = \{abab^n\} \cup \{aba^n\}$

$L = \{aba, abab, ababb, ababbb, \ldots\} \cup \{ab, aba, abaa, abaaa, \ldots\}$

$L = \{ab, aba, abab, abaa, ababb, \ldots\}$

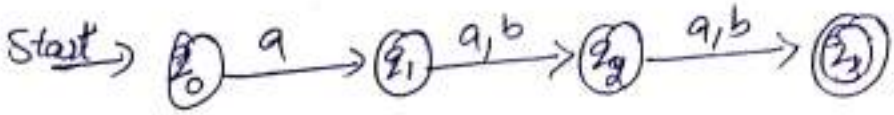




Scanned with CamScanner

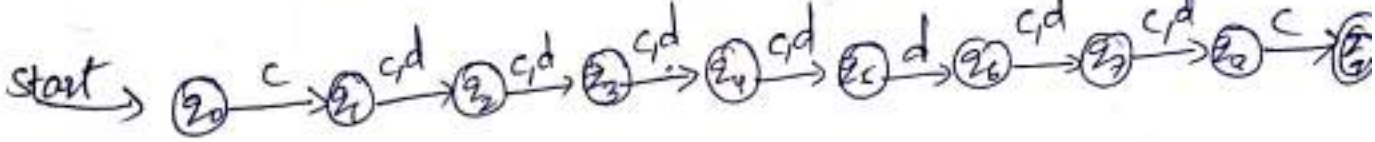
(13) Design all the string over {a,b} and 3rd symbol from right end is always 'a'.

Sol:- $L = \{aaa, abb, aab, aba, \ldots\}$



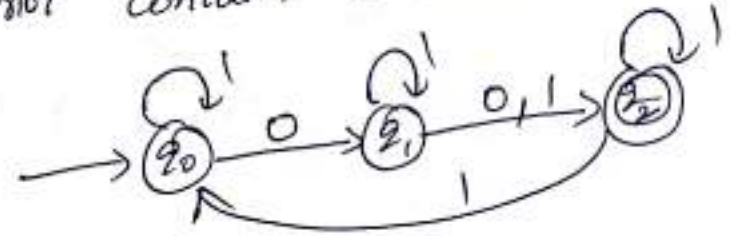
(14) Design NFA $\Sigma = \{c, d\}$ & 4th symbol from right end is D and 9th symbol the end is c.

Sol:- $L = \{cdcdddccc, \ldots\}$



(15) Design a NFA to accept set of all strings that does not contain 3 consecutive zeros.

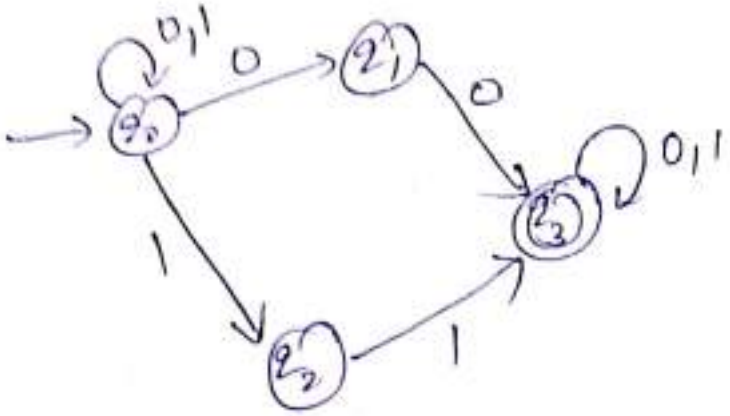
Sol:-



(16) Draw the transition diagram for a NFA which accepts all strings with either two consecutive 0's or two consecutive 1's.

Scanned with CamScanner

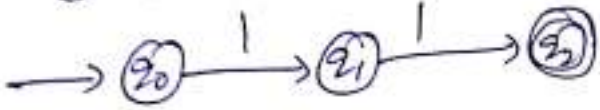
Sol:- The transition diagram with either two consecutive 0's or two consecutive 1's is



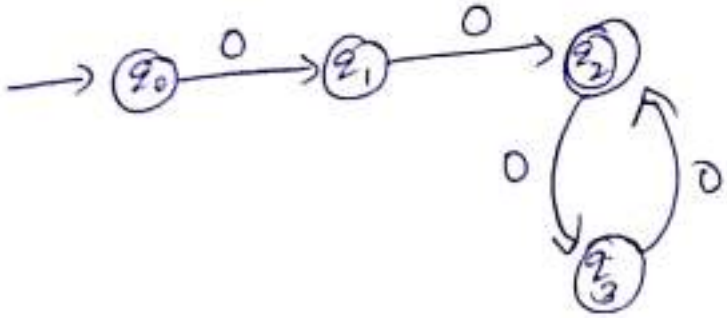
(17) Provide an NFA with at most six states for the following language:

$L = \{w / w$ Contains an even number of 0's, or exactly two 1's$\}$.

Sol:- The language may contain exactly two 1's.



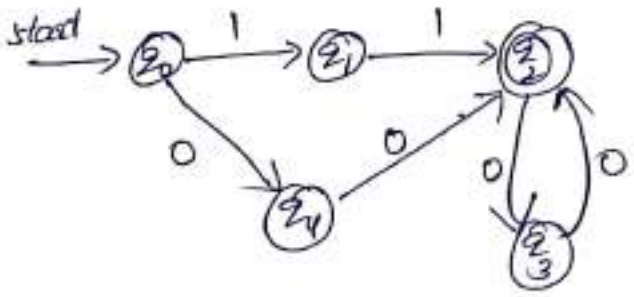
or The language may contain even number of 0's.



Scanned with CamScanner

If we combine above given transition diagrams then -

The finite automata will be

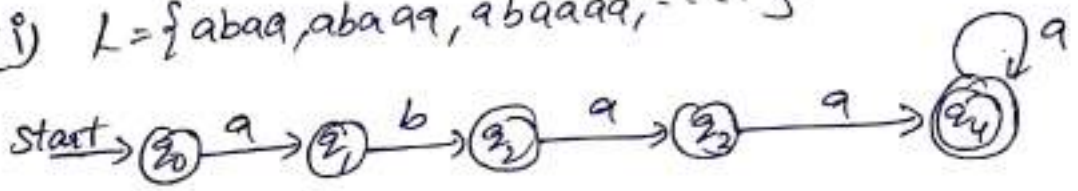


(13) Design a NFA for the following

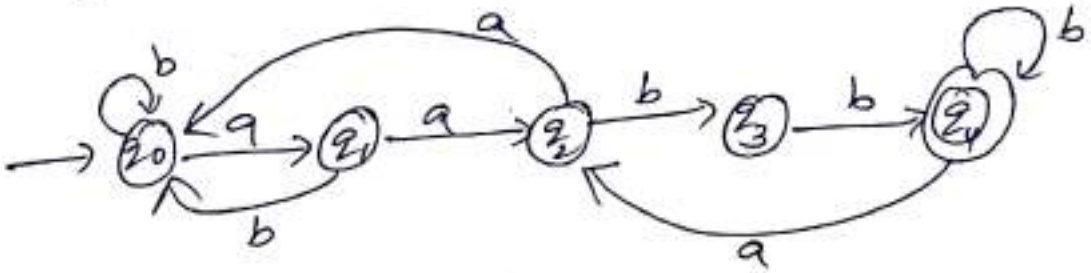
(i) $L = \{abaa^n \mid n \geq 1\}$

(ii) To accept language of all strings with 2 a's followed by 2 b's over $\{a, b\}$

Sol:- (i) $L = \{abaa, abaaa, abaaaa, \ldots\}$



(ii) $L = \{aabb, baabb, aaabb, \ldots\}$



Scanned with CamScanner

Text Search

Application of Finite Automata

① It is useful in lexical analysis process of Compilation in which tokens such as identifier, variables or constants are separated.

② It is useful in designing text editors.

③ It is useful in designing spell checkers.

④ Finite automata is useful in designing Sequential circuit

Finite automata is a very effective tool which is used in string matching & text Search algorithms.

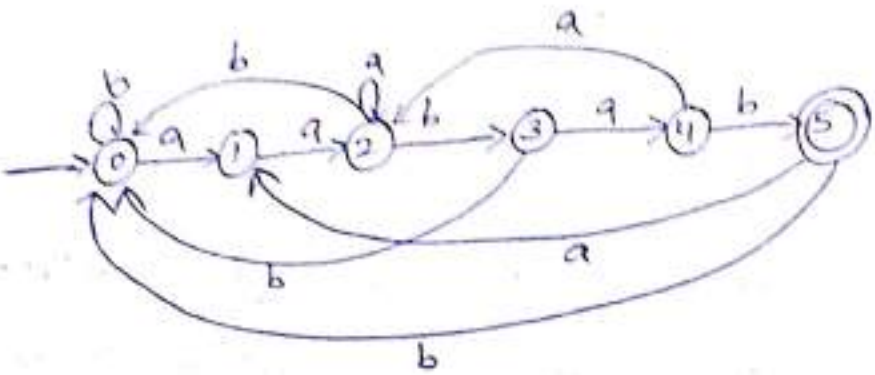
* Matching the pattern in this manner makes the process of string matching very effecient.

For example :- we can construct a finite automata for Searching the pattern P = aabab for text string -

T = aaababaabaababaab.

In above Finite Automata (FA) the dark edge going from left to right represents a spine. It corresponds to successful search of the required pattern. The edges from right to left are called failing edges. These edges are showing possible permutations of the string. Some of the failing edges are not shown.

Scanned with CamScanner



Input / State	a	b
0	1	0
1	2	—
2	2	3
3	4	0
4	2	5
5	1	0

	a	a	a	b	a	b	a	a	b	a	a	b	a	b	a	a	b
	a	a	b	a	b												
Match found →		a	a	b	a	b											
			a	a	b	a	b										
				a	a	b	a	b									
				a	a	a	b	a	b								
						a	a	b	a	b							
							a	a	b	a	b						
								a	a	b	a	b					
									a	a	b	a	b				
Match Found →										a	a	b	a	b			
											a	a	b	a	b		
												a	a	b	a	b	
													a	a	b	a	b

Thus on accepting the pattern the FA should in final state. Then it is said that pattern is searched successfully.

Scanned with CamScanner

Finite Automata with Epsilon Transitions

Importance of accepting NFA with ε transitions

As construction of DFA is very difficult for certain input set, we construct NFA. This NFA then can be converted to DFA. ε is an empty string.

* The ε-transitions are used simply to change one state to other.
* Sometimes to reach to final state we do not get proper state from start state. In such a case we simply want to reach to certain state which leads to final state. Such a transition to that specific state should be without any input symbol. Hence we require some ε-moves by which a proper state can be obtained for reaching to final state.

* Thus ε-moves play an important role in NFA.

Let $M = (Q, \Sigma, \delta, q_0, F_0)$ be a NFA with ε.

Where

Q is a finite set of states

Σ is input set

δ is a transition or a mapping function for transition from $Q \times (\Sigma \cup \epsilon)$ to 2^Q

q_0 is a start state

F is a set of final states such that $F \in Q$

Scanned with CamScanner

The string w in L accepted by NFA can be represented as, $L(M) = \{w | w \in \Sigma^*$ and δ transition for w from q_0 reaches to F$\}$

→ The ϵ-closure (p) is a set of all states which are reachable from state p on ϵ-transitions such that:

(i) ϵ-closure (p) = p where $p \in Q$

ii) If there exists ϵ-closure (p) = $\{q\}$ and $\delta(q, \epsilon) = r$ then ϵ-closure (p) = $\{q, r\}$

Steps to convert the given NFA with ϵ to NFA without ϵ.

① Find out all the transitions from each state from Q. That will be called as ϵ-closure $\{q_i\}$ where $q_i \in Q$.

② Then δ' transitions can be obtained. Then δ' transitions means an ϵ-closure on δ-moves.

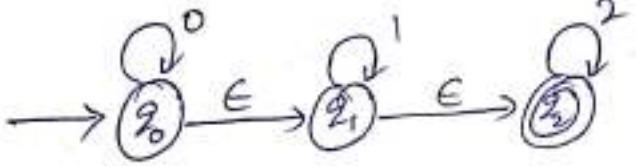
③ Step-2 is repeated for each input symbol and for each state of given NFA.

④ Using the resultant states the transition table for equivalent NFA without ϵ can be built.

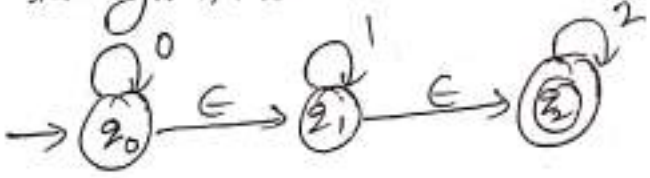
Scanned with CamScanner

① Construct NFA with ϵ which accepts a language consisting the strings of any number of 0's followed any number of 1's, followed by any number of 2's.

Ans:



② Convert the given NFA with ϵ to NFA without ϵ.



Ans:- We will first obtain ϵ-closure of each state i.e. we will find out ϵ-reachable states from current state.

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

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

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

As ϵ-closure(q_0) means with null input (no input symbol) we can reach to q_0, q_1, or q_2. In a similar manner for q_1 and q_2 ϵ-closures are obtained. Now we will obtain δ' transitions for each state on each input symbol

$$\delta'(q_0, 0) = \epsilon\text{-closure}(\delta(\delta'(q_0, \epsilon), 0))$$

$$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), 0))$$

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

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

Scanned with CamScanner

$= \epsilon\text{-closure}(q_0 \cup \phi \cup \phi)$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Scanned with CamScanner

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

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

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

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

$\delta'(q_2,0) = \phi$

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

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

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

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

$\delta'(q_2,1) = \phi$

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

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

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

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

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

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

$\delta'(q_0,2) = \{q_2\}$

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

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

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

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

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

$\delta'(q_1,2) = \{q_2\}$

Scanned with CamScanner

$\delta'(q_2, 2)$: ϵ-closure $(\delta(\delta'(q_2, \epsilon), 2))$

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

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

$= \epsilon$-closure (q_2)

$\delta'(q_2, 2) = \{q_2\}$

Now, we will summarize all the computed δ' transitions -

$\delta'(q_0, 0) = \{q_0, q_1, q_2\}$, $\delta'(q_0, 1) = \{q_1, q_2\}$, $\delta'(q_0, 2) = \{q_2\}$

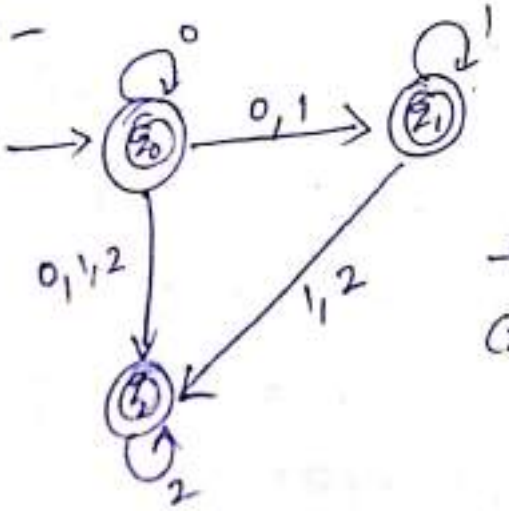
$\delta'(q_1, 0) = \phi$ $\delta'(q_1, 1) = \{q_1, q_2\}$, $\delta'(q_1, 2) = \{q_2\}$

$\delta'(q_2, 0) = \phi$ $\delta'(q_2, 1) = \phi$ $\delta'(q_2, 2) = \{q_2\}$

transition table is

State \ Input	0	1	2
→ q_0	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$

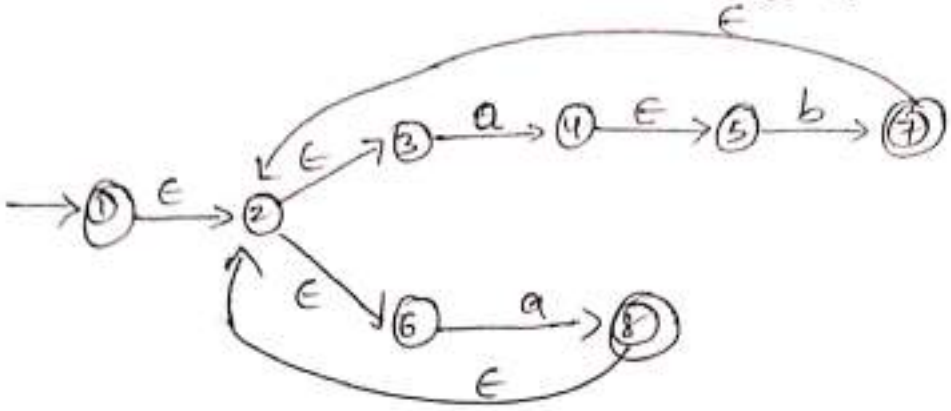
The NFA will be -



Here q_0, q_1, q_2 is a final states because ϵ-closure (q_0), q_1 & q_2 contains final state q_2.

Scanned with CamScanner

③ For the following NFA with ϵ moves convert it into an NFA without ϵ-moves and show that NFA with ϵ-moves accept the same language as shown in fig.



Ans:

ϵ-closure (1) = {1, 2, 3, 6}

ϵ-closure (2) = {2, 3, 6}

ϵ-closure (3) = {3}

ϵ-closure (4) = {4, 5}

ϵ-closure (5) = {5}

ϵ-closure (6) = {6}

ϵ-closure (7) = {2, 3, 6, 7}

ϵ-closure (8) = {2, 3, 6, 8}

Now, the input transitions on these states,

$\delta'(1, a)$ = ϵ-closure ($\delta(\delta'(1, \epsilon), a)$)

= ϵ-closure ($\delta(1, 2, 3, 6), a$)

= ϵ-closure ($\delta(1, a) \cup \delta(2, a) \cup \delta(3, a) \cup \delta(6, a)$)

= ϵ-closure (4, 8)

= {2, 3, 4, 5, 6, 8}

Scanned with CamScanner

$\delta'(1,b)$ = ϵ-closure $(\delta(\delta'(1,\epsilon),b))$

= ϵ-closure $(\delta(1,2,3,6),b)$

= ϵ-closure (ϕ)

= ϕ

$\delta'(2,a)$ = ϵ-closure $(\delta(\delta'(2,\epsilon),a))$

= ϵ-closure $(\delta(2,3,6),a)$

= ϵ-closure $(4,8)$

= $\{2,3,4,5,6,8\}$

$\delta'(2,b)$ = ϵ-closure $(\delta(\delta'(2,\epsilon),b))$

= ϵ-closure $(\delta(2,3,6),b)$

= ϵ-closure (ϕ)

= ϕ

$\delta'(3,a)$ = ϵ-closure $(\delta(\delta'(3,\epsilon),a))$

= ϵ-closure $(\delta(3,a))$

= ϵ-closure (4)

= $\{4,5\}$

$\delta'(3,b)$ = ϵ-closure $(\delta(\delta'(3,\epsilon),b))$

= ϵ-closure $(\delta(3,b))$

= ϵ-closure (ϕ)

= ϕ

$\delta'(4,a)$ = ϵ-closure $(\delta(\delta'(4,\epsilon),a))$

= ϵ-closure $(\delta(4,5),a)$

= ϵ-closure (ϕ)

= ϕ

Scanned with CamScanner

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

$= \epsilon\text{-closure}(\delta(4,5),b)$

$= \epsilon\text{-closure}(\delta(4,b) \cup \delta(5,b))$

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

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

$= \{2,3,6,7\}$

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

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

$= \phi$

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

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

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

$= \{2,3,6,7\}$

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

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

$= \epsilon\text{-closure}(8)$

$= \{2,3,6,8\}$

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

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

$= \epsilon\text{-closure}(8)$

$= \phi$

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

$= \epsilon\text{-closure}(\delta(2,3,6,7),a)$

$= \epsilon\text{-closure}(4,8) = \{2,3,4,5,6,8\}$

Scanned with CamScanner

$\delta'(7,b)$ = ϵ-closure $(\delta(\delta'(7,\epsilon),b))$

= ϵ-closure $(\delta(2,3,6,7),b)$

= ϵ-closure $(\delta(2,b) \cup \delta(3,b) \cup \delta(6,b) \cup \delta(7,b))$

= ϵ-closure $(\phi \cup \phi \cup \phi \cup \phi)$

= ϕ

$\delta'(8,a)$ = ϵ-closure $(\delta(\delta'(8,\epsilon),a))$

= ϵ-closure $(\delta(\epsilon\text{-closure}(8),a))$

= ϵ-closure $(\delta(2,3,6,8),a)$

= ϵ-closure $(\delta(2,a) \cup \delta(3,a) \cup \delta(6,a) \cup \delta(8,a))$

= ϵ-closure $(\phi \cup 4 \cup 8 \cup \phi)$

= ϵ-closure $(4,8)$ = $\{2,3,4,5,6,8\}$

$\delta'(8,b)$ = ϵ-closure $(\delta(\delta'(8,\epsilon),b))$

= ϵ-closure $(\delta(\epsilon\text{-closure}(8),b))$

= ϵ-closure $(\delta(2,3,6,8),b)$

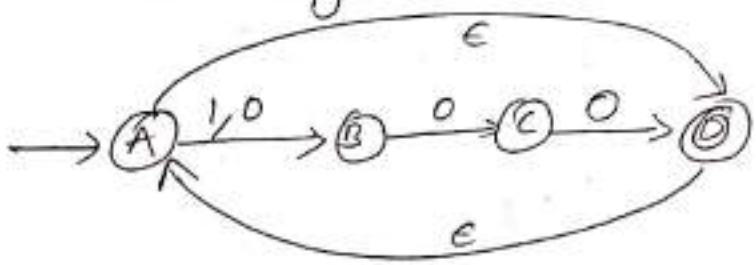
= ϵ-closure $(\phi \cup \phi \cup \phi \cup \phi) = \phi$

The transition table will be —

State \ Input	a	b
1	$\{2,3,4,5,6,8\}$	ϕ
2	$\{2,3,4,5,6,8\}$	ϕ
3	$\{4,5\}$	ϕ
4	ϕ	$\{2,3,6,7\}$
5	ϕ	$\{2,3,6,7\}$
6	$\{2,3,6,8\}$	ϕ
7	$\{2,3,4,5,6,8\}$	ϕ
8	$\{2,3,4,5,6,8\}$	ϕ

Scanned with CamScanner

④ Construct NFA for given NFA with ϵ-moves.



Ans:

ϵ-closure(A) = {A, D}

ϵ-closure(B) = {B}

ϵ-closure(C) = {C}

ϵ-closure(D) = {A, D}'

Now δ' transitions for each state on each input

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

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

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

$= \epsilon\text{-closure}(\delta(A,0) \cup \delta(D,0))$

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

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

$= \{B\}$

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

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

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

$= \epsilon\text{-closure}(\delta(A,1) \cup \delta(D,1))$

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

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

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

Scanned with CamScanner

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

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

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

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

$= \{C\}$

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

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

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

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

$= \phi$

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

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

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

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

$= \{A,D\}$

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

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

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

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

$= \phi$

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

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

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

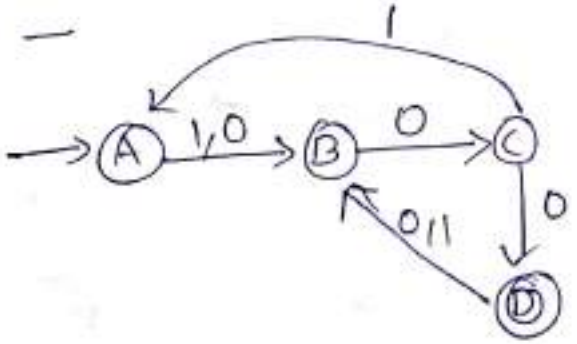
$= \epsilon\text{-closure}(B)$

$\delta'(D,0) = \{B\}$

Scanned with CamScanner

$$\delta'(D,1) = \epsilon\text{-closure}(\delta(\delta'(D,\epsilon),1))$$
$$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(D),1))$$
$$= \epsilon\text{-closure}(\delta(A,D),1)$$
$$= \epsilon\text{-closure}(\delta(A,1) \cup \delta(D,1))$$
$$= \epsilon\text{-closure}(B \cup \phi)$$
$$\delta'(D,1) = \{B\}$$

The NFA is —



Conversion of NFA to DFA

Steps: 1) The start state of NFA M will be the start for DFA M'. Hence add q_0 of NFA (start state) to Q'. Then find the transitions from this start state.

2) For each state $[q_1, q_2, q_3, \ldots q_i]$ in Q' the transitions for each input symbol Σ can be obtained as,

(i) $\delta'([q_1, q_2, \ldots q_i], a) = \delta(q_1, a) \cup \delta(q_2, a) \cup \ldots \delta(q_i, a)$

$= [q_1, q_2, \ldots q_k]$ may be some state.

(ii) Add the state $[q_1, q_2, \ldots q_k]$ to DFA if it is not already added in Q'.

Scanned with CamScanner

(iii) Then find the transitions for every input symbol from Σ for state $[q_1, q_2, \cdots q_k]$. If we get some state $[q_1, q_2, \cdots q_n]$ which is not in Q' of DFA then add this state to Q'.

iv) If there is no new state generating then stop the process after finding all the transitions.

3) For the state $[q_1, q_2, \cdots q_n] \in Q'$ of DFA if any one state q_i is a final state of NFA then $[q_1, q_2, \cdots q_n]$ becomes a final state. Thus the set of all the final states $\in F'$ of DFA.

Statement :-

Consider a NFA $M = (q_0, \Sigma, \delta, Q, F)$ then our objective is to construct an equivalent DFA M'.

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

Procedure :-

Step 1 :- Identify the states $Q' \subseteq 2^Q$ (Power set)

Step 2 :- $\Sigma' = \Sigma$

Step 3 :- $q_0' = q_0$

Step 4 :- $F' \in 2^Q$

Scanned with CamScanner

Step 5: All transitions can be identified by using transition table approach.

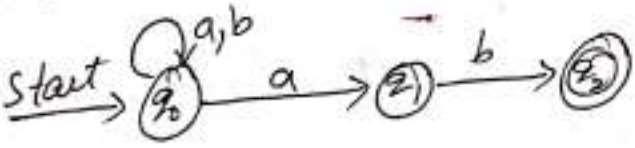
(i) Initially identify the transitions with initial state.

(ii) If any new states appears under input column these are not appeared in state column then considered that state as new state.

(iii) Repeat the above process untill no new states are found.

Step 6:- In the resultant transition table if any missing transitions (ϕ) are there then introduce dead node.

① Construct DFA for the given NFA



NFA transition table is

States	a	b
→ q_0	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
(*) q_2	ϕ	ϕ

DFA transition table is

States	a	b
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
(*) $\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$

Scanned with CamScanner

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

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

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

$= \{q_0, q_1\}$

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

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

$= q_0 \cup q_2$

$= \{q_0, q_2\}$

$\delta(\{q_0, q_2\}, a)$

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

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

$= \{q_0, q_1\}$

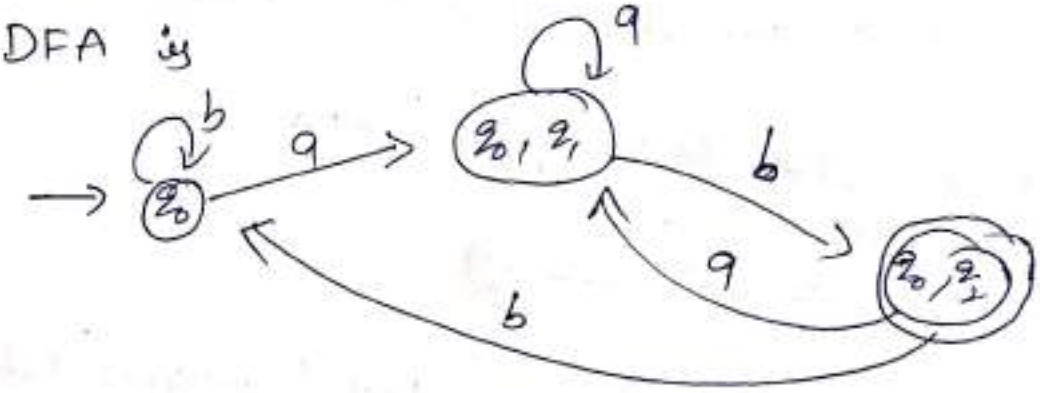
$\delta(\{q_0, q_2\}, b)$

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

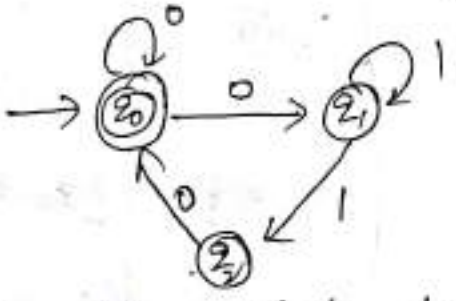
$= q_0 \cup \phi$

$= \{q_0\}$

DFA is



② Design DFA from the given NFA



NFA transition table is

state	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	ϕ
q_1	ϕ	$\{q_1, q_2\}$
q_2	q_0	ϕ

Scanned with CamScanner

DFA transition table is

State	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	ϕ
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	q_0	$\{q_1, q_2\}$

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

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

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

$= \{q_0, q_1\}$

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

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

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

$= \{q_1, q_2\}$

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

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

$= \phi \cup \{q_0\}$

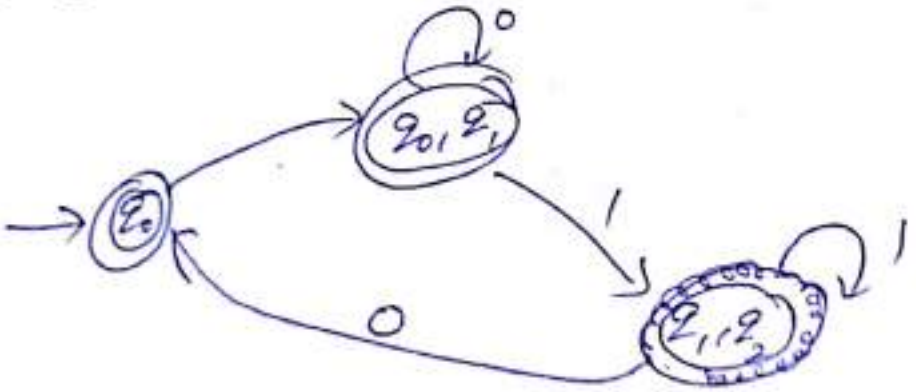
$= \{q_0\}$

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

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

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

$= \{q_1, q_2\}$



q_0 and $\{q_0, q_1\}$ are final states.

Scanned with CamScanner

③ Construct DFA for given NFA.

	0	1
→ P	{q, s}	q
q	r	{q, r}
r	s	{q, r}
s	—	{P}

DFA transition table

Ans:-

	0	1
→ P	{q, s}	q
{q, s}	r	{P, q, r}
{q}	r	{q, r}
r	s	{q, r}
{P, q, r}	{q, r, s}	{q, r}
{s}	ϕ	P
{q, r}	{r, s}	{q, r}
{q, r, s}	{r, s}	{P, q, r}
{r, s}	{s}	{P, q, r}

Scanned with CamScanner

$\delta(\{q,s\},0)$

$= \delta(q,0) \cup \delta(s,0)$

$= r \cup \phi$

$= r$

$\delta(\{q,s\},1)$

$= \delta(q,1) \cup \delta(s,1)$

$= \{q,r\} \cup \{p\}$

$= \{p,q,r\}$

$\delta(q,0) = r \qquad \delta(q,1) = \{q,r\}$

$\delta(\{p,q,r\},0)$

$= \delta(p,0) \cup \delta(q,0) \cup \delta(r,0)$

$= \{q,s\} \cup \{r\} \cup s$

$= \{q,r,s\}$

$\delta(\{p,q,r\},1)$

$= \delta(p,1) \cup \delta(q,1) \cup \delta(r,1)$

$= q \cup \{q,r\} \cup \{q,r\}$

$= \{q,r\}$

$\delta(\{q,r\},0)$

$= \delta(q,0) \cup \delta(r,0)$

$= r \cup s$

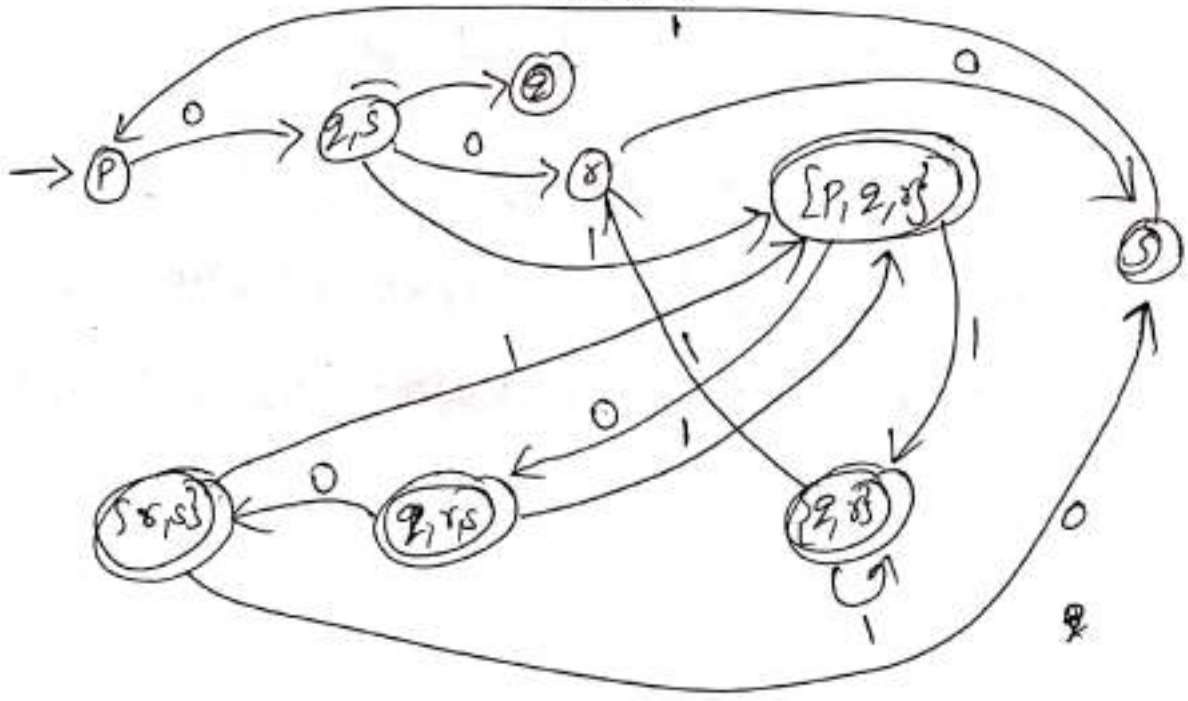
$= \{r,s\}$

$\delta(\{q,r\},1)$

$= \delta(q,1) \cup \delta(r,1)$

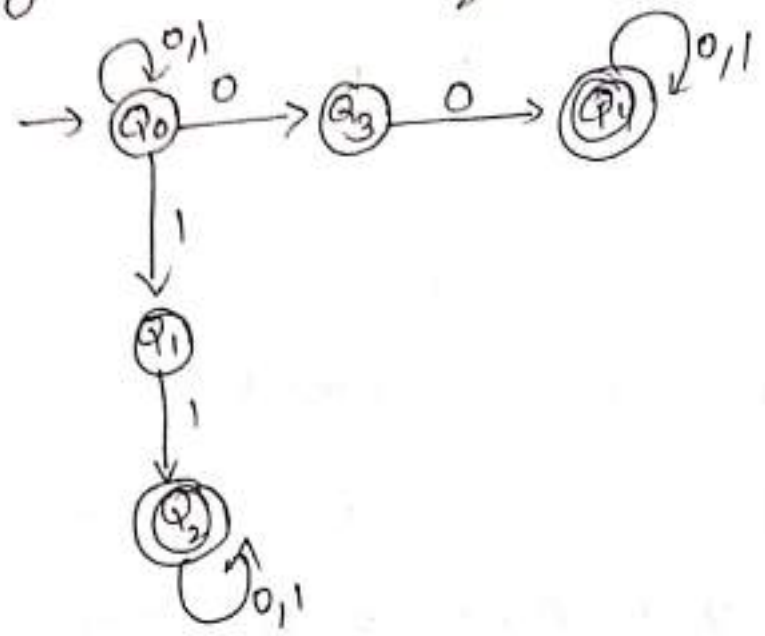
$= \{q,r\} \cup \{q,r\}$

$= \{q,r\}$



Scanned with CamScanner

(4). Convert given NFA to its equivalent DFA.



Q: Transition table for given NFA is

State \ Input	0	1
→ Q_0	$\{Q_0, Q_3\}$	$\{Q_0, Q_1\}$
Q_1	ϕ	Q_2
(Q_2)	Q_2	Q_2
Q_3	Q_4	ϕ
(Q_4)	Q_4	Q_4

we have two state $\{Q_0, Q_3\}$ and $\{Q_0, Q_1\}$

$\delta(\{Q_0, Q_3\}, 0) = \delta(Q_0, 0) \cup \delta(Q_1, 0) = \{Q_0, Q_3\}$

$\delta\{(Q_0, Q_3), 1\} = \delta(Q_0, 1) \cup \delta(Q_1, 1) = \{Q_0, Q_1\} \cup \{Q_2\}$

$= \{Q_0, Q_1, Q_2\} \rightarrow$ New state

Scanned with CamScanner

$\delta(\{Q_0, Q_3\}, 0) = \delta(Q_0, 0) \cup \delta(Q_3, 0) = \{Q_0, Q_3\} \cup \{Q_4\} = \{Q_0, Q_3, Q_4\}$ → New

$\delta(\{Q_0, Q_3\}, 1) = \delta(Q_0, 1) \cup \delta(Q_3, 1) = \{Q_0, Q_1\}$

$\delta(\{Q_0, Q_1, Q_2\}, 0) = \{Q_0, Q_2, Q_3\} = \{Q_0 Q_2 Q_3\}$ → New state

$\delta(\{Q_0, Q_1, Q_2\}, 1) = \{Q_0, Q_2, Q_3\} = \{Q_0 Q_2 Q_3\}$:

$\delta(\{Q_0 Q_3 Q_4\}, 0) = \{Q_0, Q_3, Q_4\} = \{Q_0 Q_3 Q_4\}$

$\delta((Q_0 Q_3 Q_4), 1) = \{Q_0 Q_1 Q_4\}$ → New state

$\delta(\{Q_0 Q_2 Q_3\}, 0) = \{Q_0 Q_2 Q_3 Q_4\}$ → New state

$\delta(\{Q_0, Q_2, Q_3\}, 1) = \{Q_0 Q_1 Q_2\}$

$\delta(\{Q_0, Q_1, Q_4\}, 0) = \{Q_0 Q_3 Q_4\}$

$\delta(\{Q_0 Q_1 Q_4\}, 1) = \{Q_0 Q_1 Q_2 Q_4\}$ → New state

$\delta(\{Q_0 Q_2 Q_3 Q_4\}, 0) = \{Q_0 Q_3 Q_4\}$

$\delta(\{Q_0 Q_2 Q_3 Q_4\}, 1) = \{Q_0 Q_1 Q_2 Q_4\}$

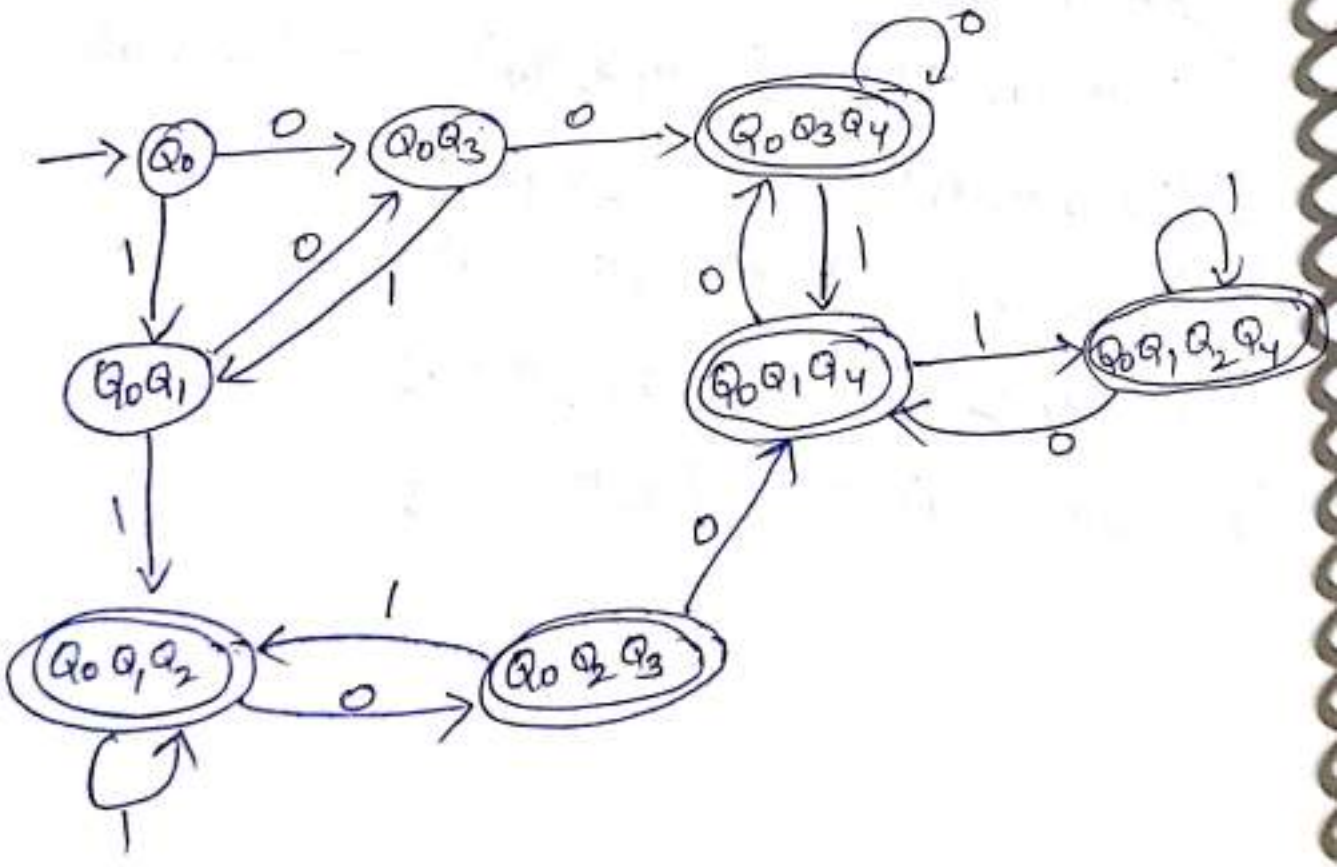
$\delta(\{Q_0 Q_1 Q_2 Q_4\}, 0) = \{Q_0 Q_2 Q_3 Q_4\}$

$\delta(\{Q_0 Q_1 Q_2 Q_4\}, 1) = \{Q_0 Q_1 Q_2 Q_4\}$

Scanned with CamScanner

DFA transition table :-

State \ Input	0	1
→ Q_0	$\{Q_0Q_3\}$	$\{Q_0Q_1\}$
$\{Q_0Q_1\}$	$\{Q_0Q_3\}$	$\{Q_0Q_1Q_2\}$
$\{Q_0Q_3\}$	$\{Q_0Q_3Q_4\}$	$\{Q_0Q_1\}$
$\{Q_0Q_1Q_2\}$	$\{Q_0Q_2Q_3\}$	$\{Q_0Q_1Q_2\}$
$\{Q_0Q_3Q_4\}$	$\{Q_0Q_3Q_4\}$	$\{Q_0Q_1Q_4\}$
$\{Q_0Q_2Q_3\}$	$\{Q_0Q_1Q_4\}$	$\{Q_0Q_1Q_2\}$
$\{Q_0Q_1Q_4\}$	$\{Q_0Q_3Q_4\}$	$\{Q_0Q_1Q_2Q_4\}$
$\{Q_0Q_1Q_2Q_4\}$	$\{Q_0Q_1Q_4\}$	$\{Q_0Q_1Q_2Q_4\}$



Scanned with CamScanner

Conversion of NFA with ϵ to DFA

Statement:- Consider a machine $M = (Q, \Sigma, \delta, q_0, F)$ with ϵ-transitions (NFA) then our objective construct an equivalent DFA $M_D = (Q_D, \Sigma_D, \delta_D, q_{0D}, F_D)$

Procedure:

Step 1 :- $Q_D \subseteq 2^Q$

Step 2 :- $\Sigma_D = \Sigma - \{\epsilon\}$

Step 3 :- Find ϵ-closure(q_0)

ϵ-closure$(q_0) = \{p_1, p_2, p_3, \dots p_4\} = A$

$A \in Q_D$, A will be act as initial state of DFA.

Step 4 :- To identify the transitions use extend transition function δ_D, Initially find transitions for Initial state of DFA.

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

Step 5 :- If any new states found we need to identify suitable transition with every input symbol.

Step 6 :- Repeat the above process until no new states are found.

Scanned with CamScanner

① Construct a DFA for the given NFA with ϵ-transitions.

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

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

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

Now we will obtain δ' transitions.

Let $\epsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}$ = "it as State A.

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

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

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

$= \{q_0, q_1, q_2\}$, i.e State A

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

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

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

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

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

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

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

$= \{q_2\} = C$

Scanned with CamScanner

$\delta'(B,0)$ = ϵ-closure $(\delta(q_1,q_2),0)$

= ϵ-closure $(\delta(q_1,0) \cup \delta(q_2,0))$

= ϵ-closure (ϕ)

= ϕ

$\delta'(B,1)$ = ϵ-closure $(\delta(q_1,q_2),1)$

= ϵ-closure $(\delta(q_1,1) \cup \delta(q_2,1))$

= ϵ-closure $\{q_1\}$

= $\{q_1,q_2\}$ i.e. state B itself.

$\delta'(B,2)$ = ϵ-closure $(\delta(q_1,q_2),2)$

= ϵ-closure $(\delta(q_1,2) \cup \delta(q_2,2))$

= ϵ-closure (q_2)

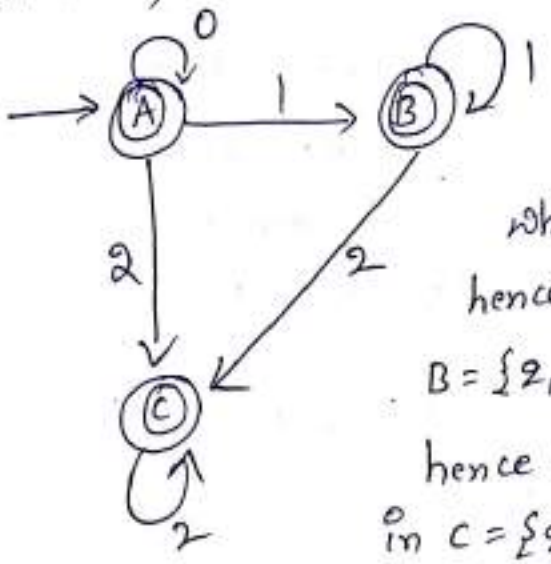
= $\{q_2\}$ i.e. state C

$\delta'(C,0)$ = ϵ-closure $(\delta(q_2,0) = \epsilon$-closure $(\phi) = \{\phi\}$

$\delta'(C,1)$ = ϵ-closure $(\delta(q_2,1) = \epsilon$-closure $(\phi) = \phi$

$\delta'(C,2)$ = ϵ-closure $(\delta(q_2,2)) = \epsilon$-closure $\{q_2\} = \{q_2\}$

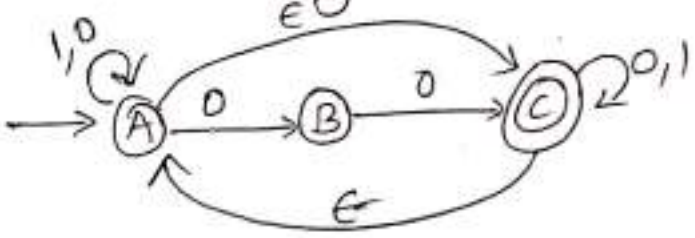
Hence the DFA is,



As $A = \{q_0,q_1,q_2\}$ in which final state q_2 lies hence A is final state in $B = \{q_1,q_2\}$ the state q_2 lies hence B is also final state in $C = \{q_2\}$, the state q_2 lies hence C is also a final state.

Scanned with CamScanner

② Construct DFA for given NFA with ε-moves



Ans:

ϵ-closure (A) = {A, C}

ϵ-closure (B) = {B}

ϵ-closure (C) = {A, C}

We will call {A, C} as state q_0. Now obtain δ transitions for q_0 state.

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

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

$= \epsilon$-closure $(\delta(A, C), 0)$

$= \epsilon$-closure $(\delta(A, 0) \cup \delta(C, 0))$

$= \epsilon$-closure (A, B, C)

$= \{A, B, C\} \longrightarrow$ New state call it as q_1

$\delta'(q_0, 1) = \epsilon$-closure $(\delta(\delta'(q_0, \epsilon), 1))$

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

$= \epsilon$-closure $(\delta(A, C), 1)$

$= \epsilon$-closure $(\delta(A, 1) \cup \delta(C, 1))$

$= \epsilon$-closure (A, C)

$= \{A, C\} \longrightarrow$ old state i.e. q_0

New state $\{A, B, C\} = q_1$ is obtained now will apply input transitions on q_1.

Scanned with CamScanner

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

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(A,B,C), 0))$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(A) \cup \epsilon\text{-closure}(B) \cup \epsilon\text{-closure}(C), 0))$

$= \epsilon\text{-closure}(\delta(A,B,C), 0)$

$= \epsilon\text{-closure}(\delta(A,0) \cup \delta(B,0) \cup \delta(C,0))$

$= \epsilon\text{-closure}(A,B,C)$

$= \{A,B,C\}$ i.e. state q_1 itself.

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

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(A,B,C), 1))$

$= \epsilon\text{-closure}(\delta(A,B,C), 1)$

$= \epsilon\text{-closure}(\delta(A,1) \cup \delta(B,1) \cup \delta(C,1))$

$= \epsilon\text{-closure}(A,C)$

$= \{A,C\}$, i.e. state q_0

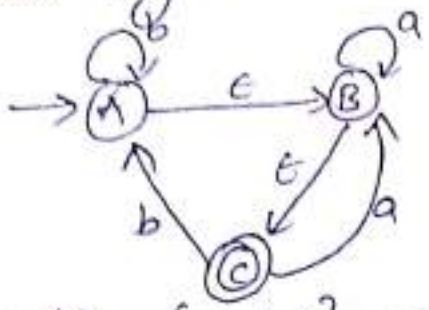
As no new state is getting formed, the DFA will be

(3) Convert the following NFA with ϵ to equivalent DFA.

	a	b	c
→A	ϕ	A	B
B	B	ϕ	C
©C	B	A	ϕ

Scanned with CamScanner

Ans:- transition diagram from the given transition table



ϵ-closure (A) = {A, B, C} ⟶ call it as q_0 state

ϵ-closure (B) = {B, C}

ϵ-closure (C) = {C}

$\delta'(q_0, a) = \epsilon\text{-closure}(\delta(A,B,C), a))$

$= \epsilon\text{-closure}(\delta(A,a) \cup \delta(B,a) \cup \delta(C,a))$

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

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

$= \{B, C\}$ ⟶ call it as state q_1

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

$\delta'(q_0, b) = \epsilon\text{-closure}(\delta(A,B,C), b))$

$= \epsilon\text{-closure}(\delta(A,b) \cup \delta(B,b) \cup \delta(C,b))$

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

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

$= \{A, B, C\}$ ⟶ old state i.e. q_0

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

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

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

$= \epsilon\text{-closure}(\delta(B,a) \cup \delta(C,a))$

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

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

$= \{B, C\}$ ⟶ i.e. q_1

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

Scanned with CamScanner

$\delta'(q_1, b) = \epsilon\text{-closure}(\delta(q_1, b))$

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

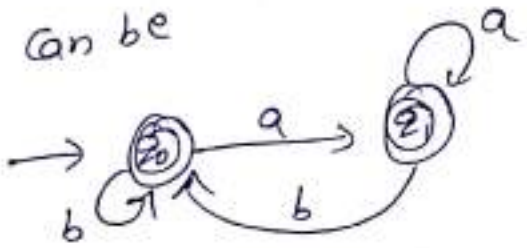
$= \epsilon\text{-closure}(\delta(B,b) \cup \delta(C,b))$

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

$= \epsilon\text{-closure}(A) = \{A, B, C\} \rightarrow$ i.e. q_0

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

The DFA can be



(4) Convert the following NFA with ϵ ~~to~~ moves to DFA.

Ans:

$\epsilon\text{-closure}(A) = \{A, B\} \longrightarrow$ call it as state q_0

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

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

$\epsilon\text{-closure}(D) = \{B, D\}$

ϵ-closure (ϕ)

$\delta'(q_0, 0) = \epsilon\text{-closure}(\delta(\delta'(q_0, \epsilon), 0))$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), 0))$

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

$= \epsilon\text{-closure}(\delta(A,0) \cup \delta(B,0))$

$= \epsilon\text{-closure}(\delta(A \cup \phi)$

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

$= \{A, B\} \rightarrow$ i.e. q_0

Scanned with CamScanner

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

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

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

$= \epsilon\text{-closure}(\delta(A, 1) \cup \delta(B, 1))$

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

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

$= \{C\} \longrightarrow$ New state q_1

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

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

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

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

$= \phi$

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

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

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

$\delta'(q_1, 1) = \{D, B\} \longrightarrow$ New state i.e. q_2

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

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

$= \epsilon\text{-closure}(\delta(B, 0) \cup \delta(D, 0))$

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

$\delta'(q_2, 0) = \phi$

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

$= \epsilon\text{-closare}(\delta(B, D), 1))$

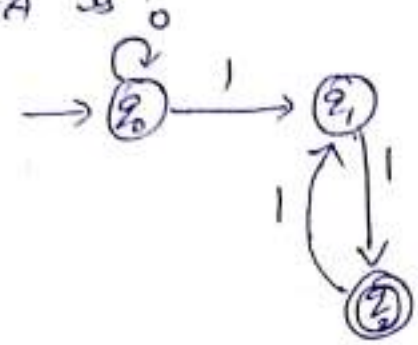
$= \epsilon\text{-closure}(\delta(B, 1) \cup \delta(D, 1))$

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

$\delta'(q_2, 1) = C \longrightarrow$ i.e. q_1

Scanned with CamScanner

The DFA is



Equivalence of 2 finite state Machines

Procedure.

Consider 2 finite state machines M_1 & M_2 then our objective to check whether they are equivalent (or) not.

Step 1: Construct Comparision table it contains n+1 columns n = input, 1 is state.

Step 2: Every state can be represented as a pair

$[q, q']$ $q \in M_1$, $q' \in M_2$

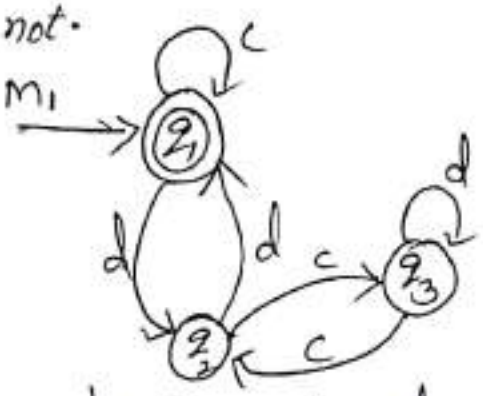
Step 3: Initially identify the transitions for q_0, q_0'

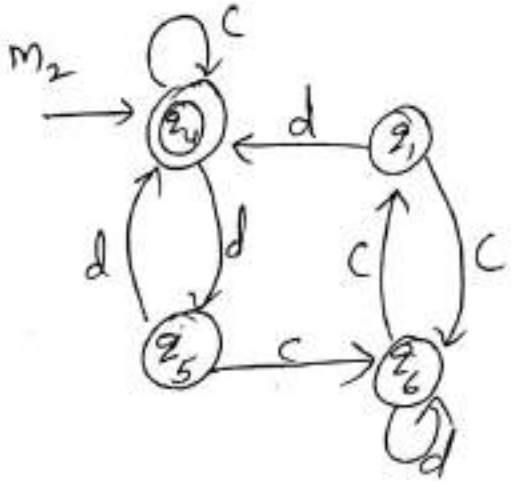
Step 4: If the resultant pair $[p, q]$ where $p \in F$, $q \notin F'$ then conclude both automata as not equivalent otherwise consider the new pair avalaible in i/p column and find the transition.

Step 5: Repeat the process until no new pairs exist.

Scanned with CamScanner

① Check whether the Finite Automata are equivalent or not.

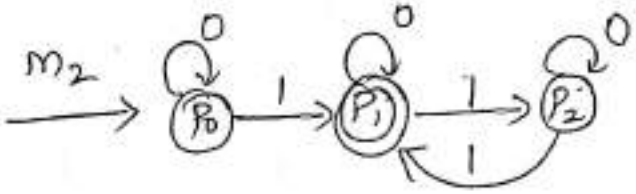




	c	d
$[q_1, q_4]$	$[q_1, q_4]$	$[q_3, q_5]$
$[q_3, q_5]$	$[q_3, q_6]$	$[q_1, q_4]$
$[q_3, q_6]$	$[q_2, q_7]$	$[q_3, q_6]$
$[q_2, q_7]$	$[q_3, q_6]$	$[q_1, q_4]$

No New pair are available, therefore both automata are equa

②

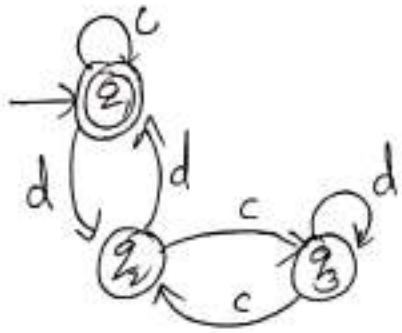


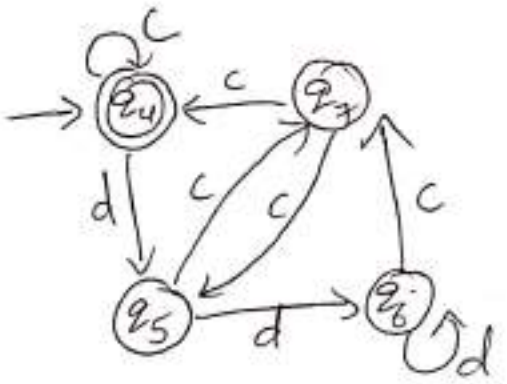
	0	1
$[q_0, P_0]$	$[q_0, P_0]$	$[q_1, P_1]$
$[q_1, P_1]$	$[q_1, P_1]$	$[q_0, P_2]$
$[q_0, P_2]$	$[q_0, P_2]$	$[q_1, P_1]$

No new pairs are available :. Both automata are equal.

Scanned with CamScanner

(3)

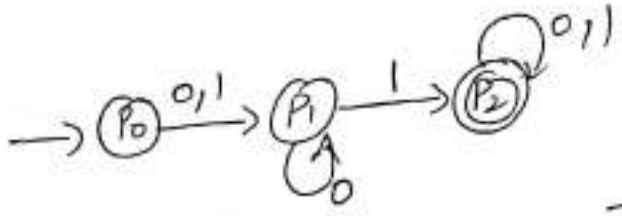


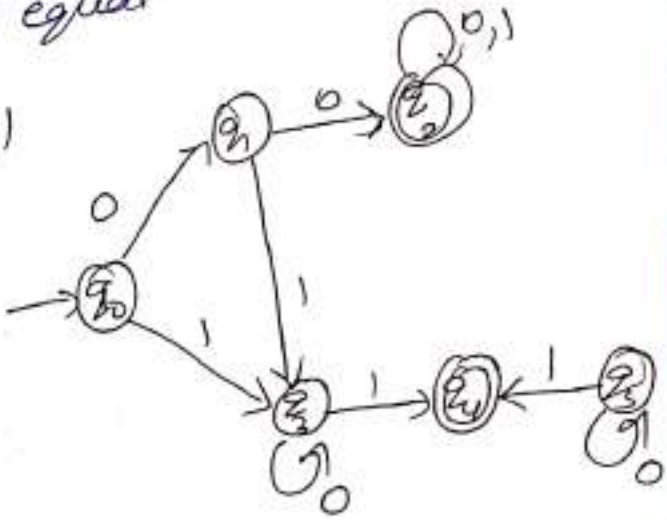


	c	d
$[q_1, q_4]$	$[q_1, q_4]$	$[q_2, q_5]$
$[q_2, q_5]$	$[q_3, q_7]$	$[q_1, q_6]$ F, 1
		X

Both automa are not equal.

(4)





	0	1
$[P_0, q_0]$	$[P_1, q_1]$	$[P_1, q_3]$
$[P_1, q_1]$	$[P_1, q_2]$	$[P_2, q_3]$
	X	

Both automata not an equivalent.

Scanned with CamScanner

Minimization of DFA

Reducing the number of states from a given F.A

Step 1: Remove all the states which are not used.

2: Draw transition table for remaining states.

3: Split the table into two tables like T_1 & T_2

T_1 contains all the final states

T_2 contains all non-final/initial states

4: Find similar rows from T_1 such that

$\delta(q, a) = p$

$\delta(r, a) = p$

- Remove one state from the similar states.

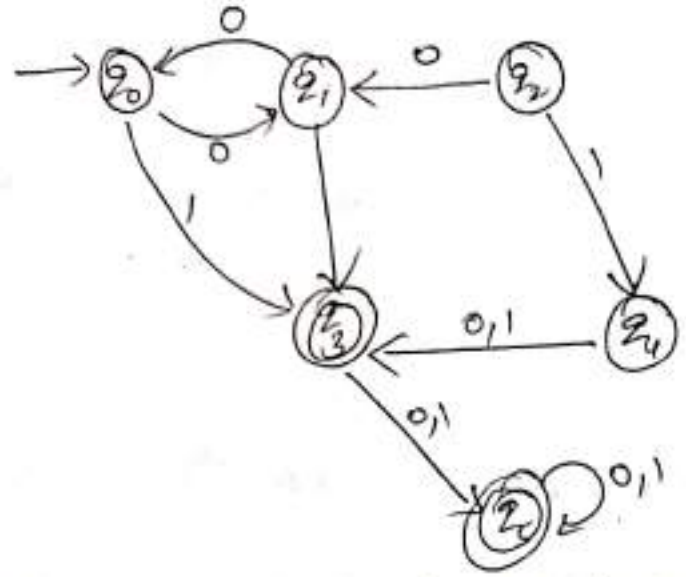
5. Repeat step 4 until we find no similar rows available in T_1

6. Repeat step 4 & step 5 for table T_2 also

7. Now combine the reduced T_1 & T_2 table. i.e. the final transition table of minimized DFA.

Scanned with CamScanner

(1).



Ans:→ Remove q_2 & q_4 in FA because q_2 & q_4 unreachable states

→ Draw transition table for the rest of the states.

States	0	1
→ q_0	q_1	q_3
q_1	q_0	q_3
q_3	q_5	q_5
q_5	q_5	q_5

→ Table which starts from non-final states

	0	1
→ q_0	q_1	q_3
q_1	q_0	q_3

Table which starts from final state

	0	1
* q_3	q_5	q_5
* q_5	q_5	q_5

Scanned with CamScanner

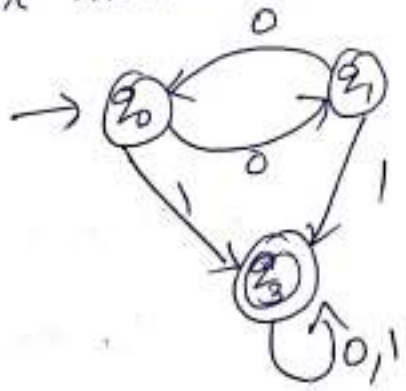
→ Table 1 has no similar rows, they will be same.
Table 2 has similar rows, so skip q_5 then replace q_5 by q_3

	0	1
q_3	q_3	q_3

→ Combine two tables

States	0	1
→ q_0	q_1	q_3
q_1	q_0	q_3
* q_3	q_3	q_3

Now, the minimized DFA is



Scanned with CamScanner

Finite Automata with output :-

→ In FA output will be expressed in the form of acceptance or rejectance.

→ Instead of generating / separating output in the form of Yes / No we need to generate some string as output, we need to maintain output along with FA

→ It can be represented in 2 ways.

1. Moore Machine
2. Mealy Machine

① **Moore Machine** :- In case of Moore Machine the concept outputs are associated with State.

It Consists of 6 tuples. $M = (Q, \Sigma, \Delta, \delta, X, q_0)$

Q — Set of State

Σ — is the finite set of symbols. called i/p alphabet

$\Delta(O)$ — is the finite set of symbols called o/p alphabet

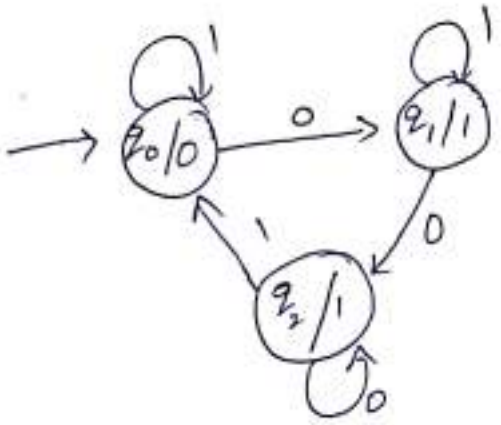
δ — is the i/p transition function where $\delta : Q \times \Sigma \to Q$

X — is the o/p transition function where $X : Q \times \Sigma \to O$

q_0 — is the initial State from where any input is processed ($q_0 \in Q$)

Scanned with CamScanner

Ex:-



	0	1	o/p
$\rightarrow q_0$	q_1	q_0	0
q_1	q_2	q_1	1
q_2	q_2	q_0	1

Consider Input w = 1, output 0.

$w = 1001$

$\delta(q_0, 1001) = 0$, $\delta(q_0, 001) = 0$

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

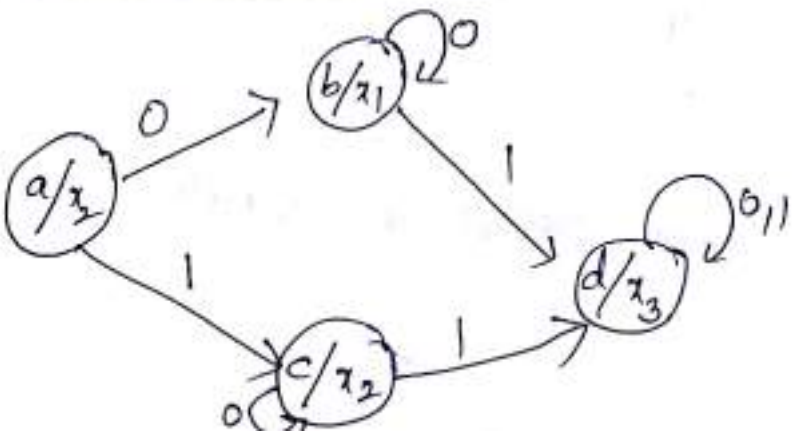
output is 0011 0

Note: In case of Moore Machine if the i/p string length is n, the output string length is "n+1".

Ex:-

	Next state		output
	I/P = 0	I/P = 1	
a	b	c	x_2
b	b	d	x_1
c	c	d	x_2
d	d	d	x_3

Moore M/c is



Scanned with CamScanner

① Construct a Moore Machine to find the complement of the given binary number.

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

If Input is 0, —— output is 1

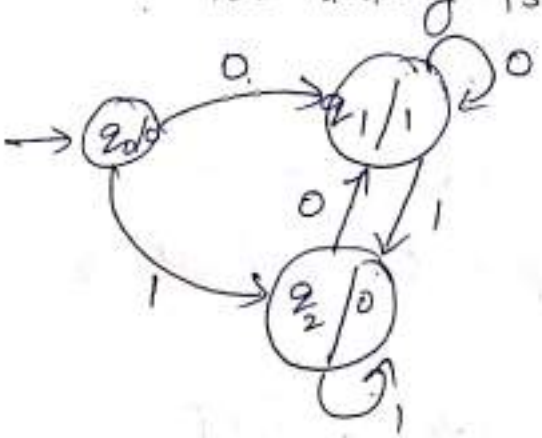
Input is 1 —— output is 0

Here, 3 states are required

one is start state

Second is for taking 0s as i/p & Produce 1 as o/p

Third is for taking 1s as i/p & Produce 0 as o/p.



current state	Next state		o/p
	0	1	
→ q_0	q_1	q_2	0
q_1	q_1	q_2	1
q_2	q_1	q_2	0

Consider the Input string is 1011

$\delta(q_0, 1011) = 0$

$\delta(q_0, 011) = 0$

$\delta(q_1, 11) = 1$

$\delta(q_2, 1) = 0$

$\delta(q_2) = 0$

The output is 00100

Scanned with CamScanner

(2) Design a Moore Machine for a binary i/p sequence such that if it has a substring 101, the machine output is A, if the i/p has substring 110 its o/p is B, otherwise its output is C.

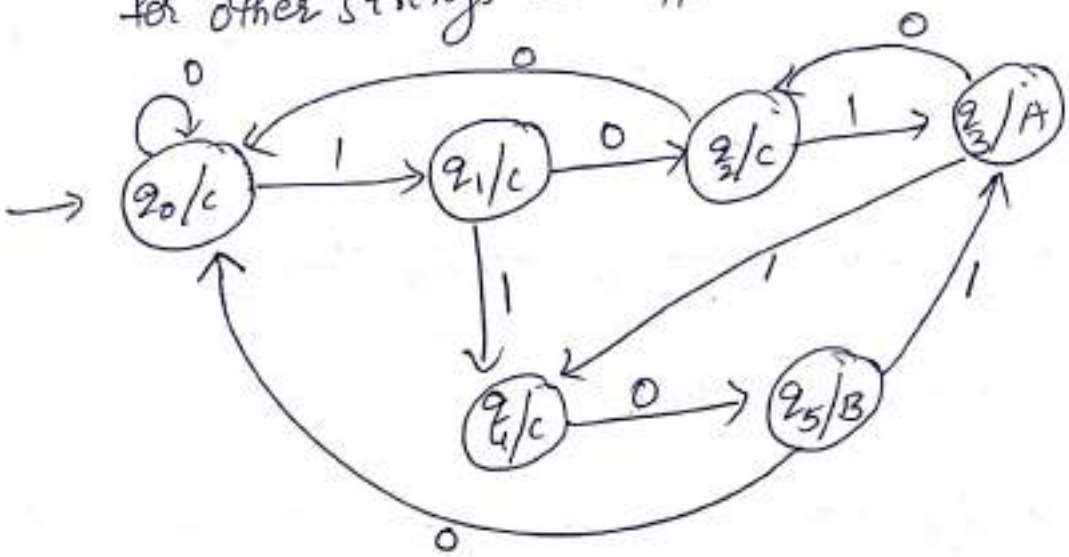
Ans: $\Sigma = \{0, 1\}$

We need to check 3 conditions

if we get i/p 101 — o/p is A

we get i/p 110 — o/p is B

for other strings — o/p is C



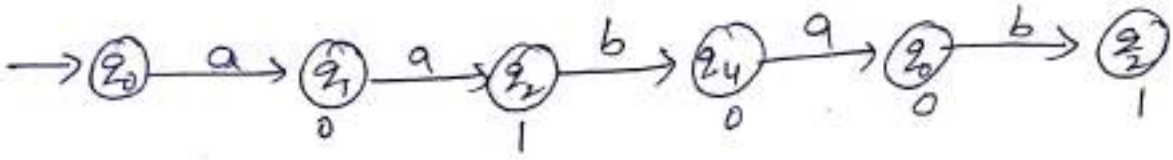
(3) Construct Moore Machine, the Input $\Sigma = (a, b)$ & output alphabet $O = (0, 1)$, Run the following i/p sequence and find the respective output.

(i) aabab (ii) abbb (iii) ababb

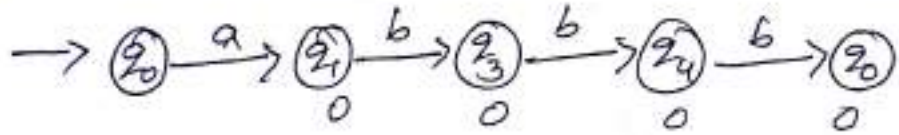
Scanned with CamScanner

State	a	b	o/p
$\rightarrow q_0$	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

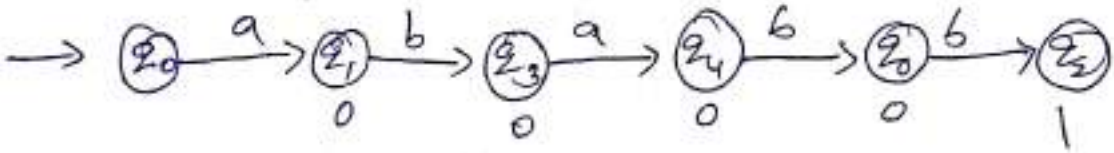
Ans:- (i) aabab = 01001



(ii) abbb = 0000



(iii) ababb = 00001



Scanned with CamScanner

Mealy Machine

Mealy machine is a finite state machine whose o/p depends on the present state as well as Present i/p.

It can be described by 6 tuples $(Q, \Sigma, O, \delta, X, q_0)$

Q is the finite set of states

Σ is the set of symbols called i/p alphabet

$\Delta - O$ is the finite set of symbols called o/p alphabet

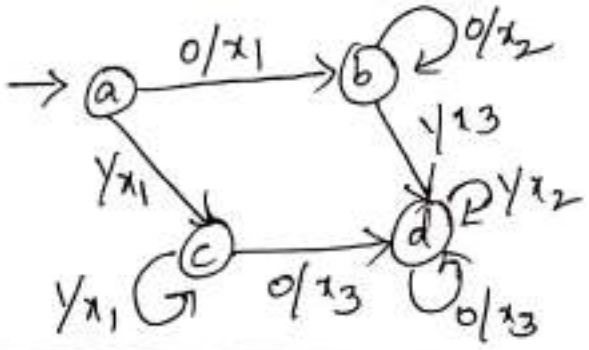
δ is the i/p transition function where $\delta: Q \times \Sigma \to Q$

X is the o/p transition function where $\delta: Q \times \Sigma \to O$

q_0 is the initial state from where any input is processed. $(q_0 \in Q)$

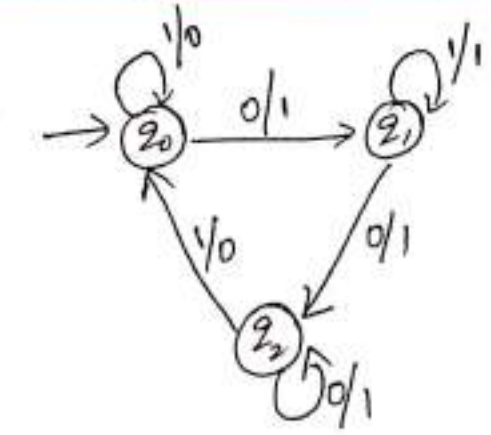
Eg:-

Present state	Next State			
	I/p = 0		I/p = 1	
	State	o/p	State	o/p
→ a	b	x_1	c	x_1
b	b	x_2	d	x_3
c	d	x_3	c	x_1
d	d	x_3	d	x_2



Scanned with CamScanner

Ex:-



	i/p 0	o/p	i/p 1	o/p
→ q_0	q_1	1	q_0	0
q_1	q_2	1	q_1	1
q_2	q_3	1	q_0	0

If $w = \epsilon$ then output $= \epsilon$

$w = 10110$

$\delta(q_0, 10110) \rightarrow 0$

$\delta(q_0, 0110) \rightarrow 1$

$\delta(q_1, 110) \rightarrow 1$

$\delta(q_1, 10) \rightarrow 1$

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

o/p → 01111

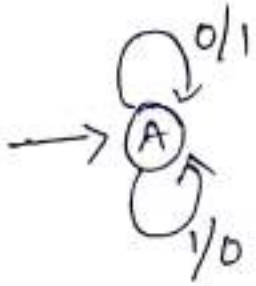
Note:- In case of mealy machine if string length is 'n' the o/p string length is "n".

Scanned with CamScanner

① Construct Mealy machine that produce 1's complement of any binary input string.

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

1's Complement means if i/p is 0 — o/p is 1
if i/p is 1 — o/p is 0



if i/p string is 101

$\delta(A, 101) = 0$

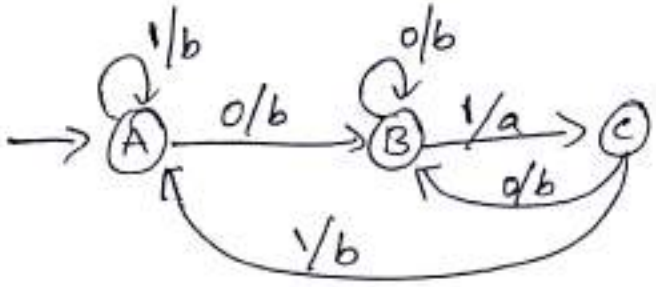
$\delta(A, 01) = 1$

$\delta(A, 1) = 0$

The output is 010

② Construct a Mealy machine that print 'a' whenever the sequence '01' is encountered in any i/p binary string.

Ans: input $\Sigma = \{0, 1\}$, output $O = \{a, b\}$

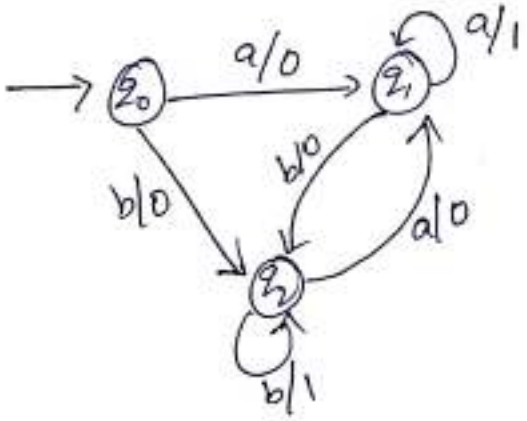


Scanned with CamScanner

③ Construct a mealy machine that accept language consists of strings from Σ^*, where $\Sigma = \{a, b\}$ and strings should end with either aa or bb.

Ans: o/p $O = \{0, 1\}$, i/p $\Sigma = \{a, b\}$

Assume that string ends with aa or bb generate o/p = 1
else generate o/p = 0



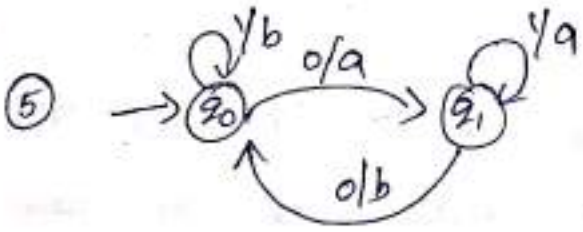
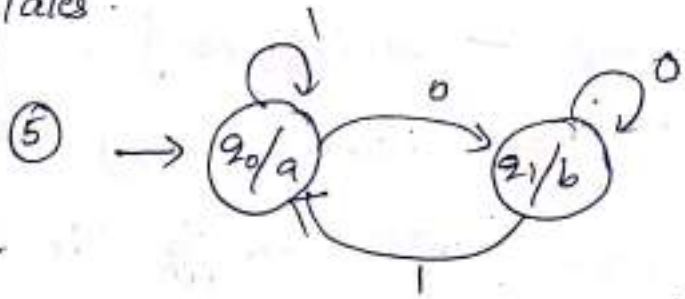
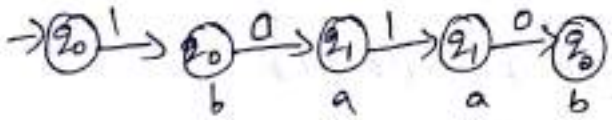
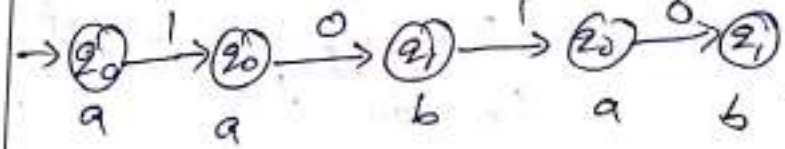
If i/p = baabb

→ • $\delta(q_0, baabb) = 0$
$\delta(q_2, aabb) = 0$
$\delta(q_1, abb) = 1$
$\delta(q_1, bb) = 0$
$\delta(q_2, b) = 1$

o/p is 00101.

Scanned with CamScanner

Difference between Mealy and Moore Machine

Mealy Machine	Moore Machine.
① Output depends both upon the Present State and Present i/p	① output depends only upon the present state.
② Generally it has (less) fewer states than moore machine.	② Generally it has more states than mealy machine.
③ The value of the output function is a function of the transition and the changes, when the input logic in the present state done.	③ The value of output function is a function of the current state the changes at the clock edges whenever state change occur.
④ It react faster to input then generally react in the same clock cycle.	④ More logic is required to decode the output resulting in more circuit delays. They generally react one clock cycle lates.
⑤ 	⑤ 
i/p - 1010 — o/p is baab	i/p - 1010 — o/p - aabab
	

Scanned with CamScanner

Conversion from Mealy to Moore Machine

Steps

① Each state (Q_j) calculate no. of different o/p that are available in the transition table in mealy machine.

② Copy the state (Q_i) of all the o/p of Q_j are same break (Q_i) into n states Q in

③ If the o/p of initial state is '0', insert new initial state at the starting which gives 1 as o/p.

→ Consider a Mealy machine $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ our objective is construct an equivalent Moore machines.

$M' = (Q \times \Delta, \Sigma, \Delta', \delta', \lambda', [q_0, b_0])$ when b_0 is a output symbol from Δ. , $b_0 \in \Delta$

Step 1:- Identify no. of states required to construct a Moore machine, every state is represent as pair state.

Step 2:- Identify the transitions δ'

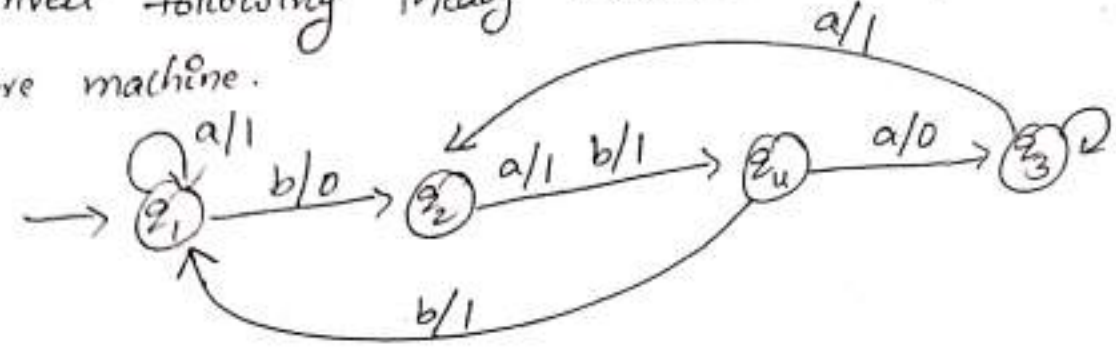
$$\delta'([q,b], a) = [\delta(q,a), \lambda(q,a)]$$

Step 3:- Identify the output symbols using output function λ'

$$\lambda'[(q,b)] = b$$

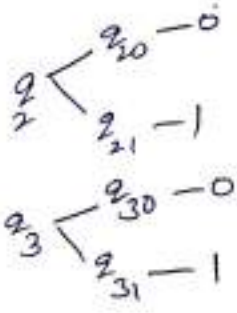
Scanned with CamScanner

① Convert following Mealy machine into equivalent Moore machine.



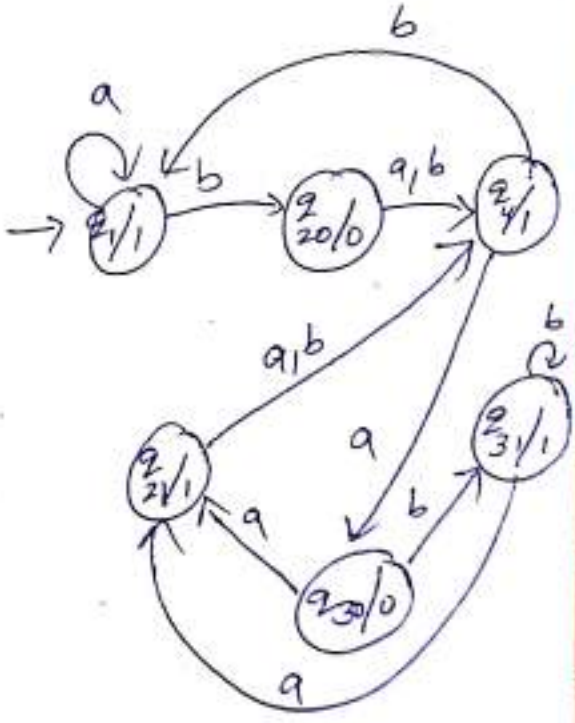
Sol: Transition table for Mealy M/C

Present State	I/p a	O/p	I/p b	O/p
→ q_1	q_1	1	q_2	0
q_2	q_4	1	q_4	1
q_3	q_2	1	q_3	1
q_4	q_3	0	q_1	1



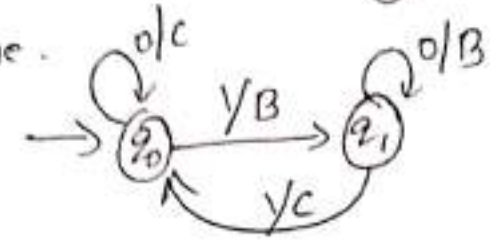
Transition table for Moore M/C

Present State	a	b	O/p
→ q_1	q_1	q_{20}	1
q_{20}	q_4	q_4	0
q_{21}	q_4	q_4	1
q_{30}	q_{21}	q_{31}	0
q_{31}	q_{21}	q_{31}	1
q_4	q_{30}	q_1	1



Scanned with CamScanner

(2) Convert Constructed Moore M/c for the given Mealy Machine.



Sol: Q = no. of states = $\{q_0, q_1\}$ $\Sigma = \{0,1\}$, $\Delta = \{B, C\}$

$\delta'([q_0,B),0) = [\delta(q_0,0), X(q_0,0)]$
$= [q_0, C]$

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

$\delta'([q_0,C],0) = [\delta(q_0,0), X(q_0,0)]$
$= [q_0, C]$

$\delta'[[q_0,C],1] = [\delta(q_0,1), X(q_0,1)]$
$= [q_1, B]$

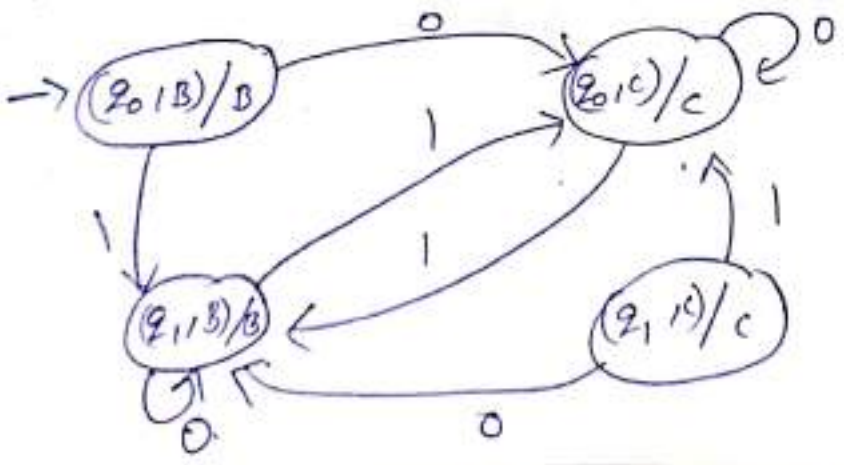
$\delta'[[q_1,B],0] = [\delta(q_1,0), X(q_1,0)] = [q_1, B]$

$\delta'([q_1,B],1] = [\delta(q_1,1), X(q_1,1)] = [q_0, C]$

$\delta'([q_1,C),0] = [\delta(q_1,0), X(q_1,0)] = [q_1, B]$

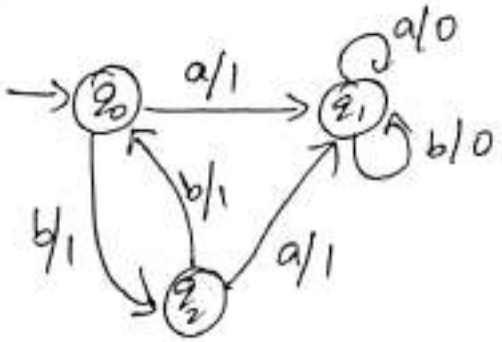
$\delta'((q_1,C),1] = [\delta(q_1,1), X(q_1,1)] = [q_0, C]$

$X'([q_0,B]) = B$, $X'(q_0,C) = C$, $X'(q_1,B) = B$, $X'(q_1,C) = C$



Scanned with CamScanner

② Construct Moore Machine for given Mealy Machine



Sol:- $\delta'((q_0,0),a) = [\delta[q_0,a), X(q_0,a)] = [q_1,1]$

$\delta'((q_0,1),b] = [\delta(q_0,b), X(q_0,b)] = [q_2,1]$

$\delta'((q_1,0),a] = [\delta(q_1,a), X(q_1,a)] = [q_1,0]$

$\delta'((q_1,1),b] = [\delta[(q_1,b), X(q_1,b)] = [q_1,0]$

$\delta'([q_2,0],a) = [\delta((q_2,a), X(q_2,a)] = [q_1,1]$

$\delta'([q_2,1),b] = [\delta(q_2,b), X(q_2,b)] = [q_0,1]$

$\delta'([q_0,0],b] = [\delta(q_0,b), X(q_0,b)] = [q_2,1]$

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

$\delta'((q_1,0),b) = [\delta(q_1,b), X(q_1,b)] = [q_1,0]$

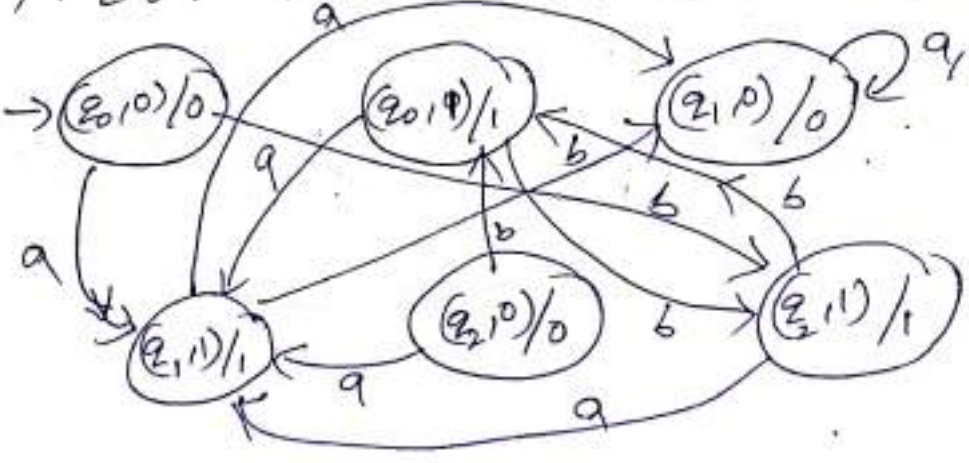
$\delta'[(q_1,1),a] = [\delta(q_1,a), X(q_1,a)] = [q_1,0]$

$\delta'((q_2,0),b] = [\delta(q_2,b), X(q_2,b)] = [q_0,1]$

$\delta'((q_2,1),a) = [\delta(q_2,a), X[q_2,a]] = [q_1,1]$

$X'[q_0,0]=0$, $X'[q_0,1)=1$, $X'(q_1,0)=0$, $X'(q_1,1)=1$, $X'(q_2,0)=0$,

$X'[q_2,1] = 1$



Scanned with CamScanner

Moorey Machine to Mealy Machine Conversion

Consider a Moore machine $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ then our objective is construct an equivalent Mealy machine

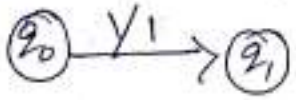
$M' = (Q, \Sigma, \Delta, \delta, \lambda', q_0)$

the output function λ' can be represented by

$$\lambda'(q, a) = \lambda(\delta(q, a))$$

Ex :-

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

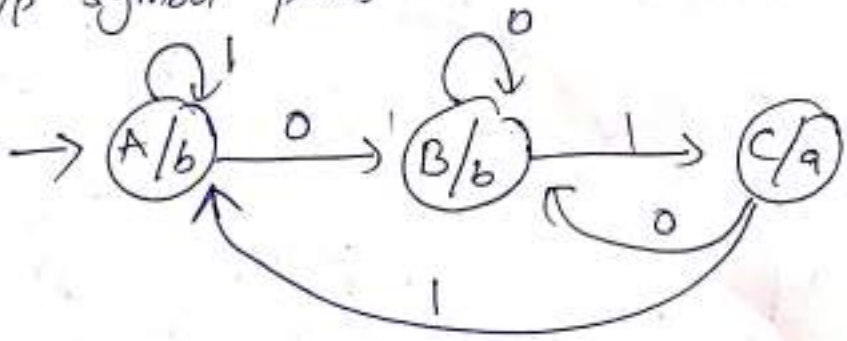


If there is no incoming transition to the state then remove the output symbol from the state.

① Construct Mealy machine from the given moore machine.

Sol :- To Convert state o/p symbols are distributed into i/p symbol paths

Moore

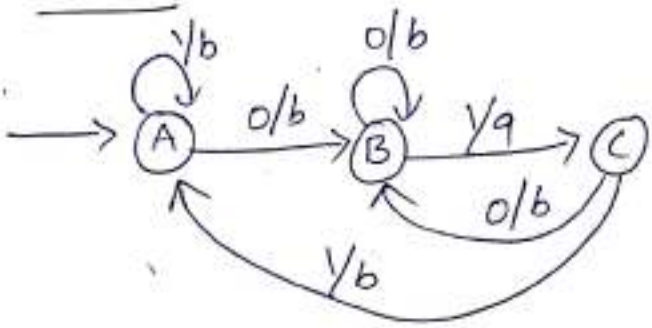


Scanned with CamScanner

State table for Moore m/c

State	0	1	o/p
→ A	B	A	b
B	B	C	b
C	B	A	a

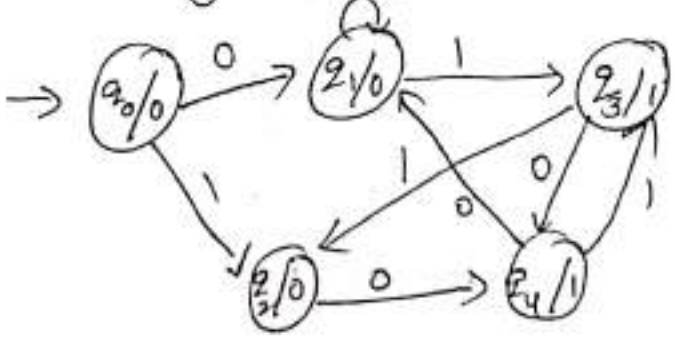
Mealy Machine



(1) Mealy m/c table

State	0		1	
	State	o/p	State	o/p
→ A	B	b	A	b
B	B	b	C	a
C	B	b	A	b

(2) Convert into Mealy m/c from Moore m/c

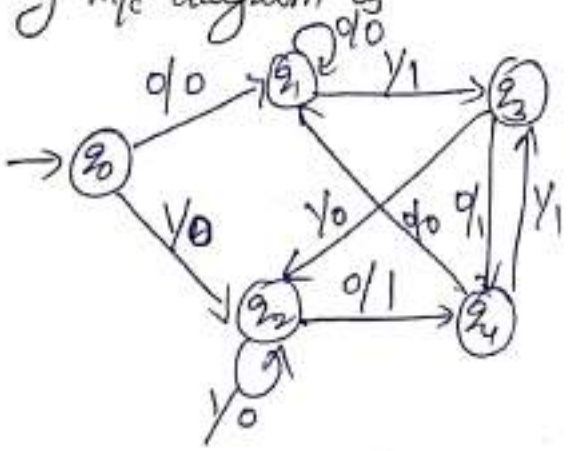


Scanned with CamScanner

Sol: Moore m/c state table is

State	0	1	o/p
→ q_0	q_1	q_2	0
q_1	q_1	q_3	0
q_2	q_4	q_2	0
q_3	q_4	q_2	1
q_4	q_1	q_3	1

Mealy m/c diagram is



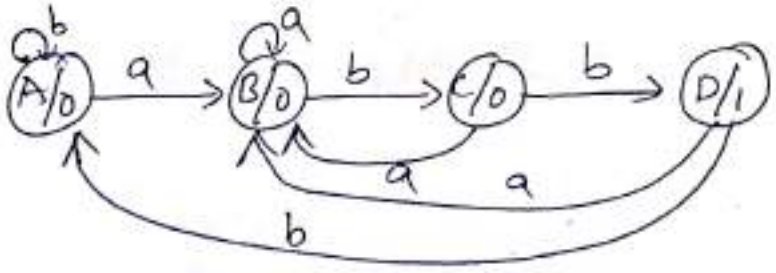
⇒

Present State	0		1	
	State	o/p	State	o/p
→ q_0	q_1	0	q_2	0
q_1	q_1	0	q_3	1
q_2	q_4	1	q_2	0
q_3	q_4	1	q_2	0
q_4	q_1	0	q_3	1

(3) The given Moore m/c counts the occurences of sequences 'abb' in any i/p binary strings over {a, b}, convert it to its equivalent Mealy m/c.

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

Moore →



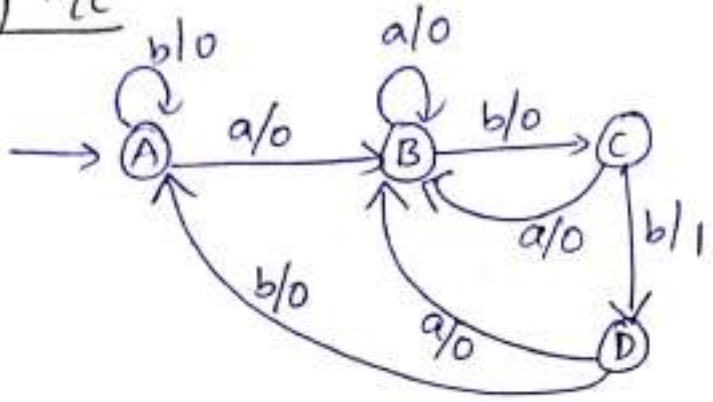
Scanned with CamScanner

Transition table is:

Moore

State	a	b	o/p
→ A	B	A	0
B	B	C	0
C	B	D	0
D	B	A	1

Mealy M/c



Mealy Transition table is

State	0		1	
	state	o/p	state	o/p
→ A	B	0	A	0
B	B	0	C	0
C	B	0	D	1
D	B	0	A	0

Scanned with CamScanner