Basic used in Theory of Computation:

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

- **Alphabet:** An alphabet is a finite, non-empty set of symbols or characters. We use the symbol Σ for an alphabet. Ex → If Σ is an alphabet containing all the 26 characters used in English lang. then Σ is finite & non-empty set and is represented as $\Sigma = \{a, b, c --- z\}$. Another example is a binary alphabet $\Sigma = \{0, 1\}$.
  $X = \{1, 2, 3 ---\}$ This is not alphabet bcz this is infinite.
  $B = \{\ \}$ This is not alphabet bcz this is empty.

- **String or word:** A string is a finite sequence of symbols choosen from some fixed alphabet. Ex → "abc" is a string over an alphabet $\Sigma = \{a, b, c --- z\}$. Another example is 01101 is a string from the binary alphabet $\Sigma = \{0, 1\}$

▲ The empty string or null string is denoted by ε. The empty string is the string with zero occurences of symbols

- **Length of a string:** The length of a string is the no. of symbols or characters in that string. If w is a string then its length is denoted by $|w|$. Eg: (1) If $w = abcd$, then length of w is $|w| = 4$.

(2) If $w = 123$, then length of w is $|w| = 3$.

(3) ε is the empty string and has length 0.

Concatenation of strings:

If W_1 and W_2 are two strings then concatenation of W with W_1 is a string and it is denoted by W_1W_2. In other words, we say that W_1 is followed by W_2 and length is $|W_1W_2| = |W_1| + |W_2|$. for example-

If $W_1 = abc$ and $W_2 = xyz$ $\quad |W_1| = 3$

then $W_1W_2 = abcxyz$ $\quad |W_2| = 3$

$|W_1W_2| = 3 + 3 = 6$.

Reverse of string:

R o s can be achieved by flipping over last symbols

If $W = abc$

then $W^R = cba$, W^R is a reverse of W

$|W| = |W^R|$.

Language: A language 'L' denotes a set of string over ~~set~~ certain alphabet. The lang ϕ is undefine as similar to infinity which is undefined & ~~a~~ this is similar to an empty box. $\{\varepsilon\}$ i.e, a lang. without an string. for example:

1) $L_1 = \{ab, aabb, aaabbb\}$ is a lang over alphabet $\Sigma = \{a, b\}$

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

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

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

5) Lang. of all string consists of 'n' 0's followed by 'n' 1's for some $n \geq 0$.

$L = \{\varepsilon, 01, 0011, 000111 \ldots\}$.

6) The set of strings over 0's and 1's with equal no. of each.

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

Note: $\phi \neq \{\varepsilon\}$.

The former has ~~one~~ no string & later has one string.

If Σ is an alphabet set. then the

Concatenation of language:

Concatenation of lang. can be defined as :- If L_1 & L_2 are two lang then their concatenation is $L = L_1 \cdot L_2$ where $L = \{w \mid w = xy ; x \in L_1 , y \in L_2\}$ i.e, the string in L_1L_2 formed by choosing a string L_1 and followed by a string in L_2 in all possible combination.

Kleene closure (or Reflexive Transitive closure):

let Σ be some alphabet then Kleene closure of Σ is denoted by Σ^* also known as RT closure. The length of Kleene closure of any alphabet is infinite. It is defined as follows -

$\Sigma^* = \{$set of all words over $\Sigma\}$.

$= \{$word of length zero, words of length one, words of length two - - $\} = \bigcup_{k}^{\infty} \Sigma^k$

$= \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 - -$

for example - -

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

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

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

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

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

2) If $S = \{a\}$. then

$S^* = \{\varepsilon, a, aa, aaa, aaaa, - - \}$.

Positive closure:

If Σ is an alphabet then positive closure of Σ is denoted by Σ^+ & defined as follows

$\Sigma^+ = \Sigma^* - \{\varepsilon\}$.

$= \{$set of all words over Σ excluding empty string $\varepsilon\}$.

for example –

(1) = If $\Sigma = \{a\}$ then $\Sigma^+ = \{a, aa, aaa, aaaa \ldots\}$.

(2) = If $\Sigma = \{0, 1\}$ then $\Sigma^+ = \{0, 1, 00, 11, 01, 10 \ldots\}$.

Formal Languages :

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

GRAMMAR :-

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

Example – $S \rightarrow 1S \mid 0S$

$S \rightarrow 0 \mid 1$

Here terminal symbols are 0 and 1. The non-terminal symbol is S.

$S \rightarrow 1S$

$\rightarrow 10S$

$\rightarrow 101S$

$\rightarrow 1010$

$\rightarrow 1011$

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

What is Automata ?

The term "Automata" is derived from the Greek Word "AUTOMATA" which means "self-acting". An automaton (Automata in plural) is an abstract is self propelled computing device which follows a pre-defined sequence of operations automatically.

An automata is finite no. of states is c/d finite Automata (FA) or finite state m/c (FSM).

Description of Finite Automata:

A finite Automata consist of a finite memory c/d input ~~take~~ tape, a read only head and finite control. The I/p is provided on the tape and head reads one symbol on the tape and move forward. The movement is decided by the finite control when input is exhausted the finite decides the Validity (~~for~~ or acceptibility) of the ~~accept~~ input by acceptance or rejection (yes/No). It does not Write anything which is a limitation of this model. ~~The~~ ~~A FA~~ model of FA is shown below-

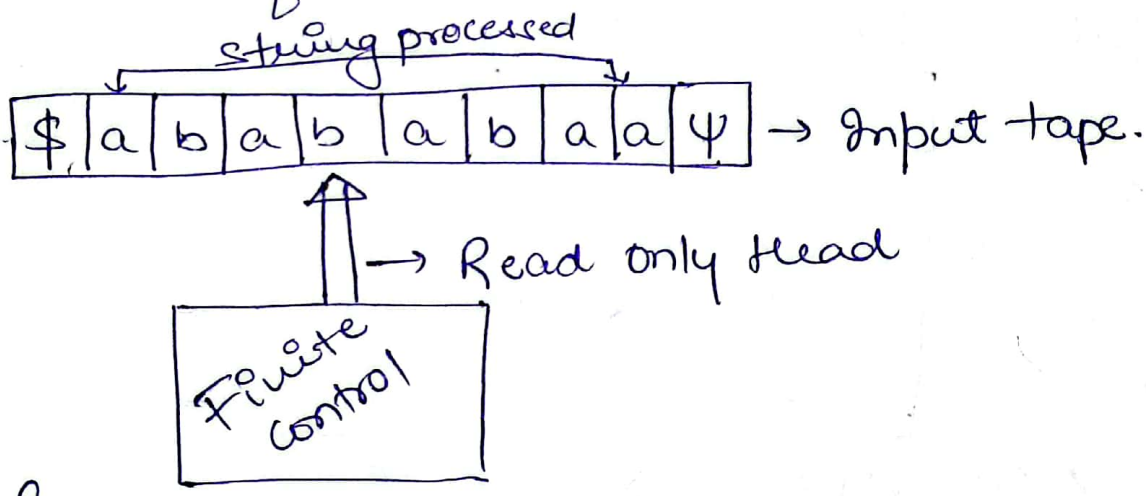


fig: Model of FA.

The input tape is divided into cells & each cell contain one symbol from the input alphabet Σ. The symbol dollar is used at the left most cell & the symbol ψ is used at the right most cell. to indicate the beginning and end of the input tape. The head reads one symbol on the input tape and finite control controls the next configuration (state). The head cannot

write and cannot move backward once the input symbol is has been read. so finite Automata cannot remember its previous reads symbols. This is one of the major limitation of FA, which shows that it cannot be used in such computation where knowledge of previous input is necessary. FA is also known as FSM or FSA

Types of FAutomata.

There are two types of FA.

1) Deterministic FA
2) Non-Deterministic FA.

Que Draw the Transition diagram for Transition Table shown below.

Soln

$\delta \mid \Sigma$	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_2	q_0
$*q_2$	q_2	q_2

Fig: Transition Table

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

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

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

$q_0 = \{q_0\}$

$\delta(q_0, a) = q_1$

$\delta(q_0, b) = q_2$

$\delta(q_1, a) = q_2$

$\delta(q_1, b) = q_0$

$\delta(q_2, a) = q_2$

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

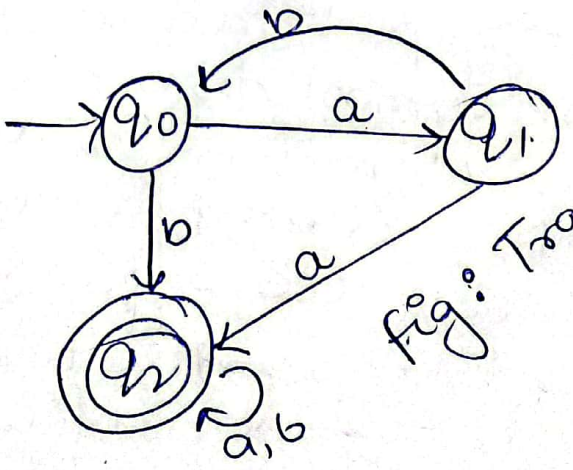
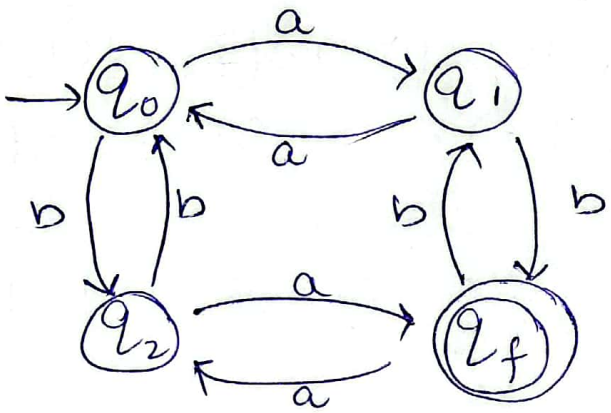


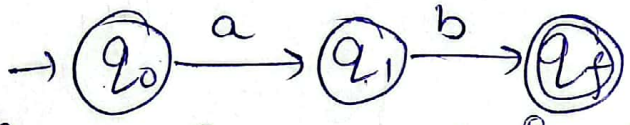
Fig: Transition Diagram

Acceptibility of a string by DFA:

Let a DFA $M = (Q, \Sigma, \delta, q_0, F)$ and input string $w \in \Sigma^*$. The string w is accepted by DFA M if and only if $\delta(q_0, w) = q_f$ where $q_f \in F$. In case w is accepted by M, the execution of string w ends in a final state. Consider the DFA shown in figure and input strings are ① ab ② abb check the acceptibility of each string.

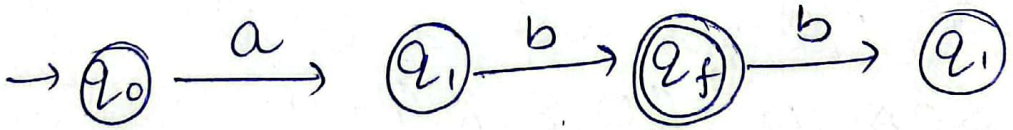


(1) Solution: let string $w_1 = ab$, then the transition sequence is as follows:



Execution ends in final state q_f, hence string "ab" is accepted.

(2) Solution: let string $w_2 = abb$, then the transition sequence is as follows:



Execution ends not in final state q_f, hence string "abb" is not accepted.

Que Consider the Dfinite Automata, $M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, q_0\}$ and the transition table is –

Sol^n

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

1) find the acceptability of the following string

(a) 110101

(b) 101100

(a) let $w_1 = 110101$

$\delta(q_0, w_1) \Rightarrow \delta(q_0, 110101) \Rightarrow \delta(q_1, 10101) \Rightarrow \delta(q_0, 0101)$

$\Rightarrow \delta(q_2, 101) \Rightarrow \delta(q_3, 01) \Rightarrow \delta(q_1, 1) \Rightarrow q_0$

It means that the string 110101 accepted.

(b) let $w_2 = 101100$

$\delta(q_0, w_2) \Rightarrow \delta(q_0, 101100) \Rightarrow \delta(q_1, 01100) \Rightarrow \delta(q_3, 1100)$

$\Rightarrow \delta(q_2, 100) \Rightarrow \delta(q_3, 00) \Rightarrow \delta(q_1, 0) \Rightarrow q_3$.

This is not accepted 101100 string.

The language of DFA:

Ques Design a finite Automata that accept set of string such that every string end in 00 over alphabet $\Sigma = (0, 1)$

Sol^n let DFA, $M = (Q, \Sigma, \delta, q_0, F)$

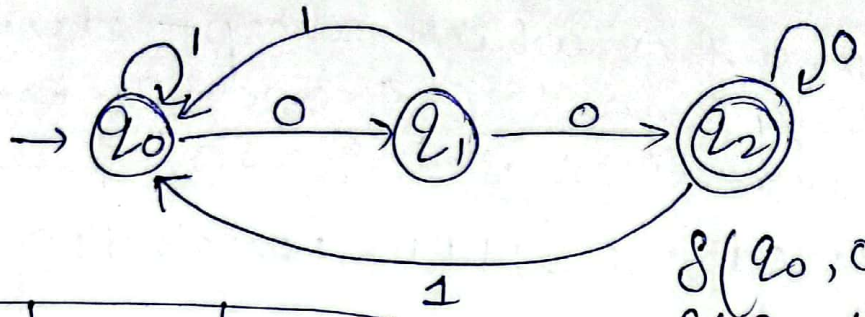
$q_0 = \{q_0\}$ (initial State)

$\Sigma = \{0, 1\}$ is given.

$L = \{00, 100, 0100, 0110100, 110100, --\}$

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

$F = \{q_2\}$

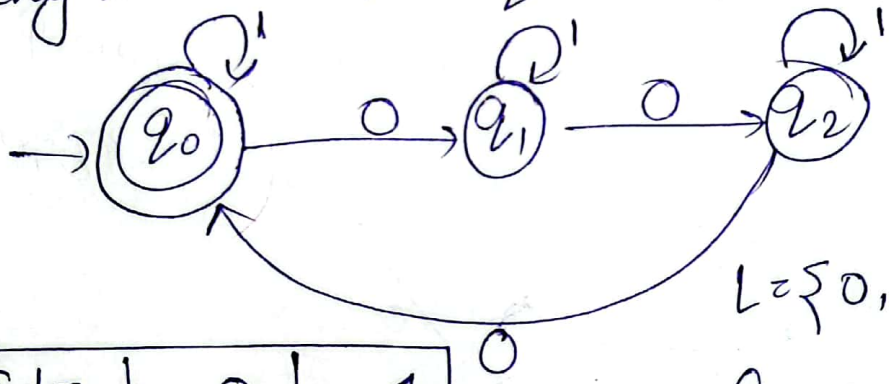


δ / Σ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
$*q_2$	q_2	q_0

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

Ques Draw a finite Automata that accepts set of string where the no. of zero in every string is multiple of 3 over alphabet $\Sigma = \{0, 1\}$

Sol^n



$L = \{0, 000, 1000, 0100 - \}$

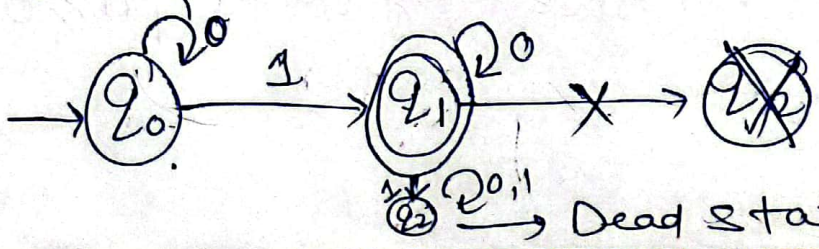
δ / Σ	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_1
$*q_2$	q_0	q_2

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

Ques Design a F.A that accepts strings containing exactly one over alphabet $\Sigma = \{0, 1\}$.

$L = \{01, 001, 1, 100, 0100, - - \}$

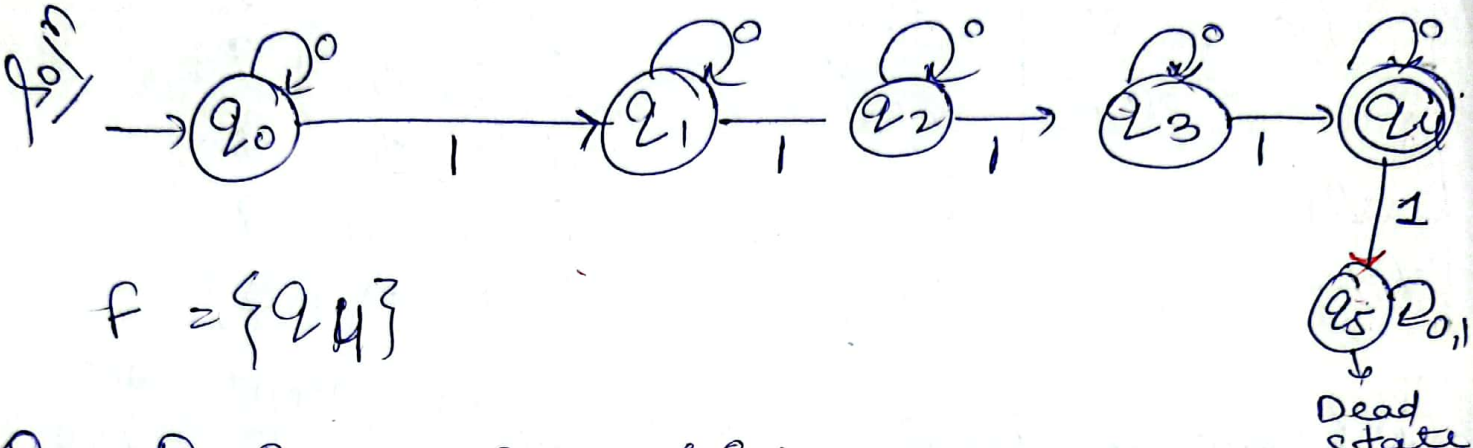
Sol^n



Ques Design a f.A which accept strings containing exactly 4 ones in every string over alphabet $\Sigma = \{0,1\}$.

$L = \{1111, 11011, 001111, 10101010, \ldots\}$

Sol^n

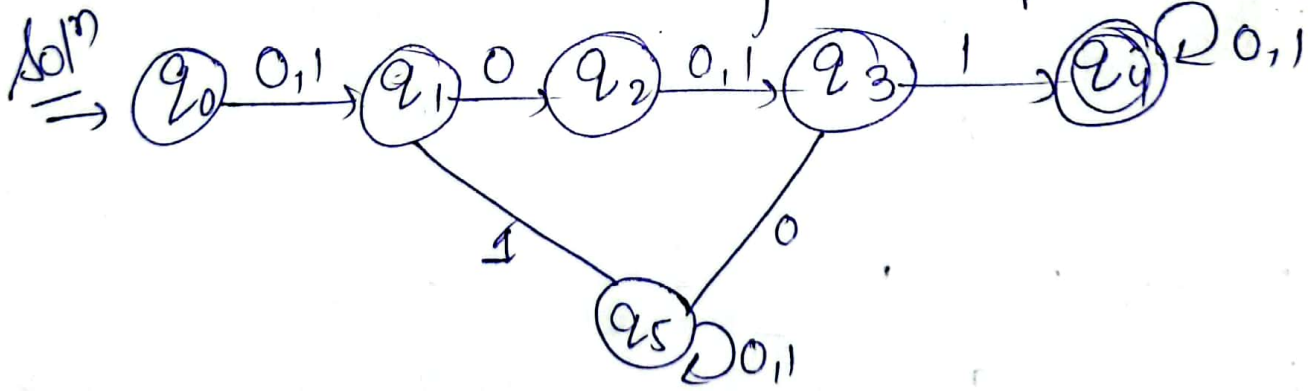


$F = \{q_4\}$

Ques Design a FA which accept the lang.

$L = \{w \in (0,1)^* \mid$ second symbol of w is 0 and fourth symbol is $1\}$.

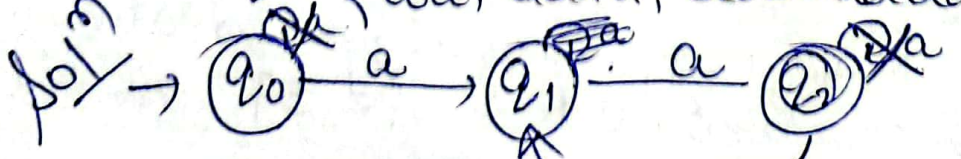
Sol^n



Ques Draw a FA for the lang $L = \{a^{2n} \mid n \geq 1\}$.

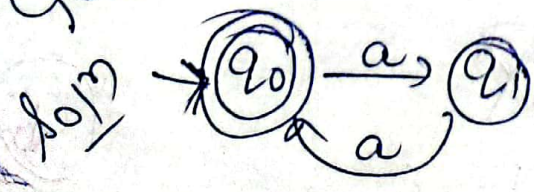
$L = \{aa, aaaa, aaaaaaaa \ldots\}$

Sol^n



Ques Draw a FA for the lang $L = \{a^{2n} \mid n \geq 0\}$

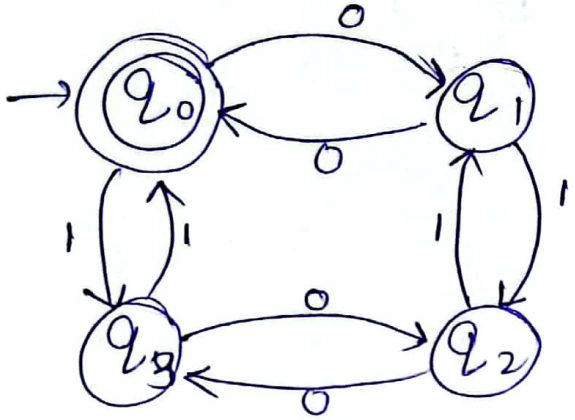
Sol^n



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

① Even no. of 1's and even no. of 0's

Soln $L = \{0101, 1010 - - \}$.

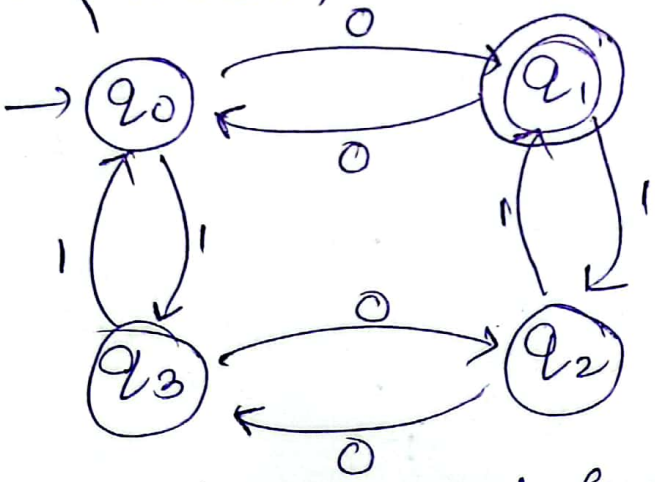


Accepted.

② Even no. of 1's and odd no. of 0's.

$L = \{11000, 10100, 110, 101 - \}$.

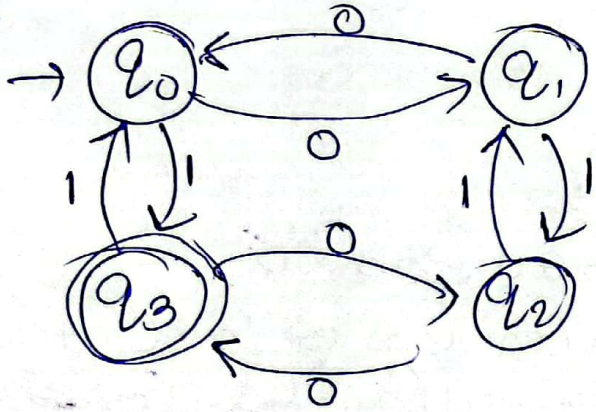
Soln



③ Odd no. of 1's and even no. of 0's

$L = \{00111, 0101100 - - \}$

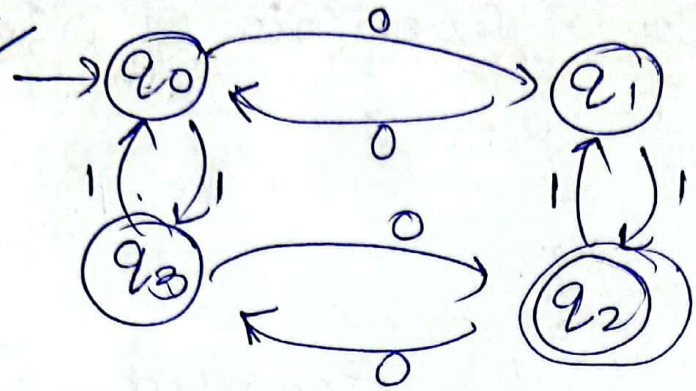
Soln



(4) Odd no. of 1's and odd no. of 0's.

$L = \{011100\}$

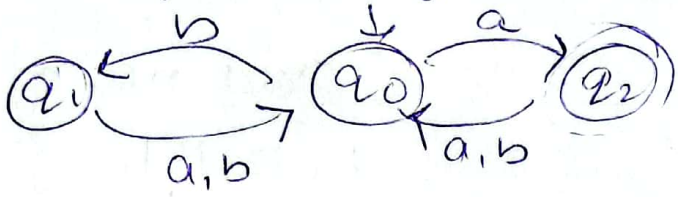
Sol^n



Ques Describe in english ~~the~~ lang. accepted by the following DFA.

$L = \{a, aaa, baa, bbaa\}$

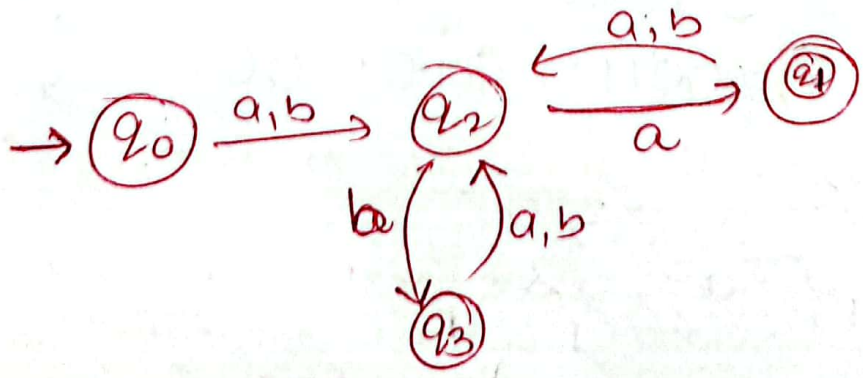
Sol^n



The lang. consisting of all the strings that are of odd length and ending with a over alphabet $\Sigma = \{a, b\}$

Ques Describe in english lang accepted by the following DFA.

Sol^n



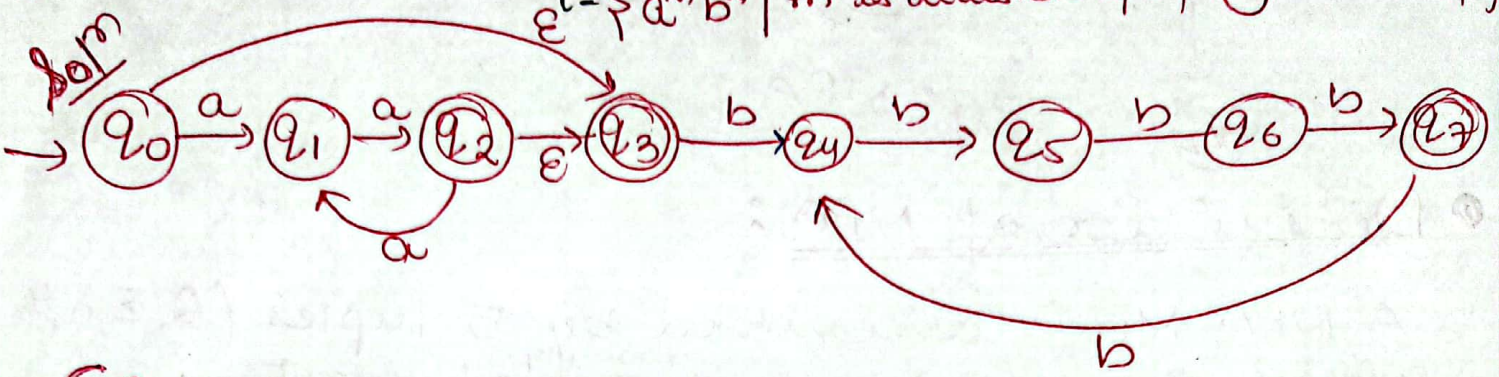
$L = \{aa, ba, abaa, bbaa, --\}$

The language consisting of all the strings that are of even length and ~~consisting~~ ending with a over alphabet $\Sigma = \{a, b\}$.

Ques Construct a DFA for the following lang.

$L = \{a^m b^n \mid m$ is divisible by 2 & n is divisible by 4$\}$

Soln

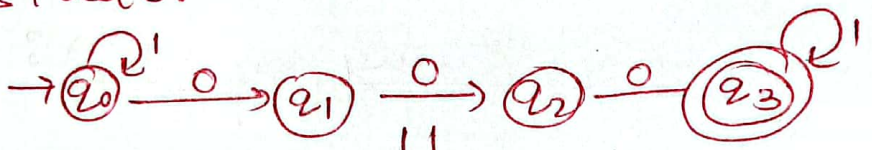


Ques Given a DFA, m suggest a procedure to draw DFA which accept the complement of the lang acceptible by m.

Procedure: ① change the final state to non-final state.

② change the non-final state to final state.

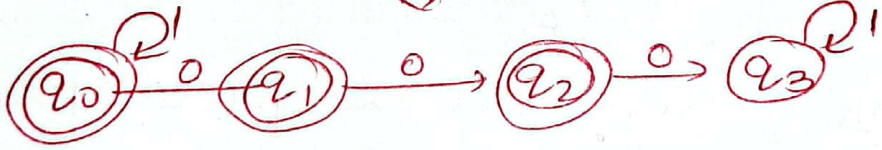
(i)



M

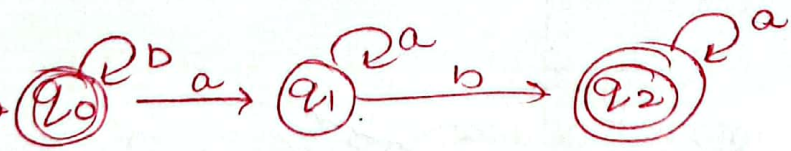
⇓

Ans



M'

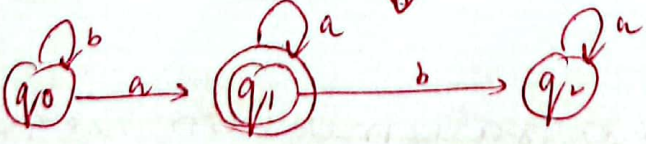
(ii)



M

⇓

Ans



M'

Non-deterministic Finite Automata:

The concept of NFA is exactly reverse of DFA. The finite Automata is called NFA when ∃ many path for a specific input from current state to next state.

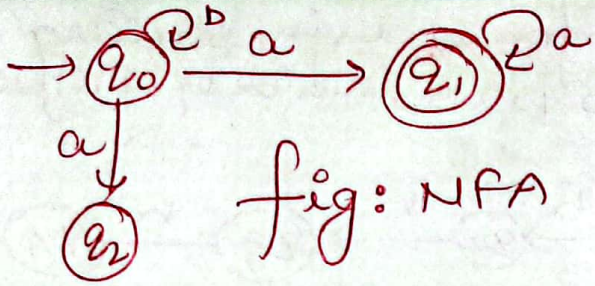


fig: NFA

• Definition of NFA :-

A NFA M_1 is described by 5 tuples $(Q, \Sigma, \delta, q_0, F)$ where Q is finite and nonempty set of state. It contains all the states, Σ is finite and non empty set of input alphabets, δ is transition fun which maps $Q \times \Sigma \to 2^Q$ q_0 belongs to Q is the initial state or starting state and F is subset of Q $(F \subseteq Q)$ is a set of final state.

• Acceptibility of a String by NFA :

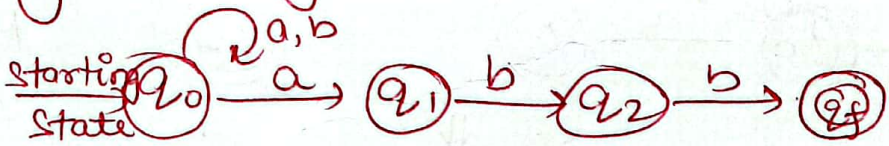
$$\delta(q_0, w) = q_f$$

for example: check the acceptibility of following string -

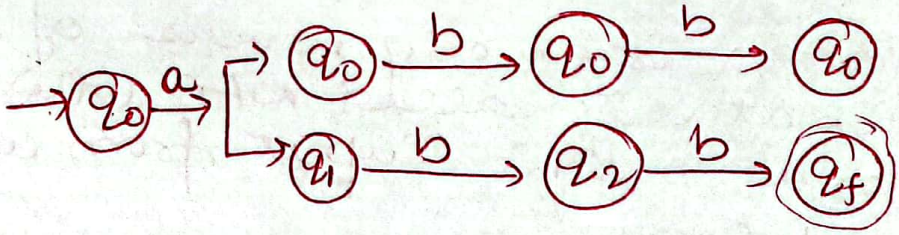
(a) abb

(b) aba

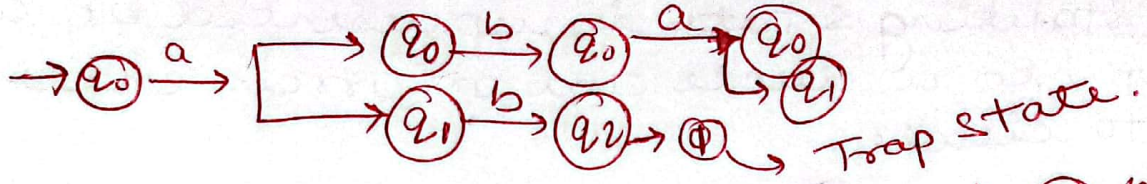
(c) abbabb



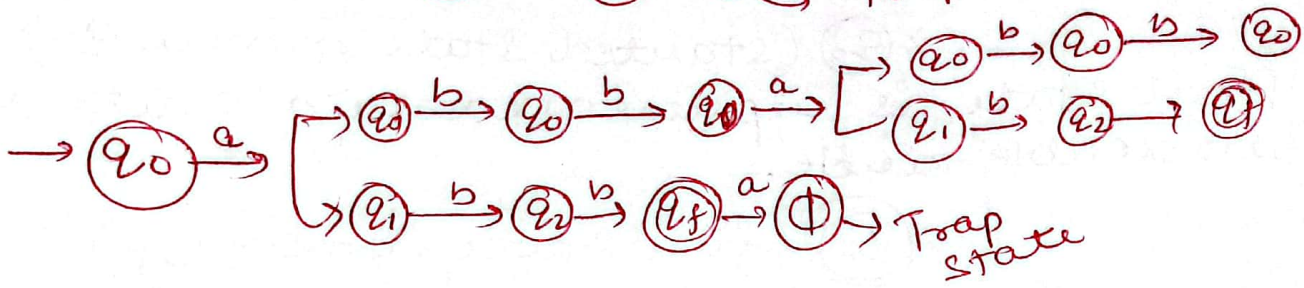
1) let w_1 = abb, then the transmission sequence for the input string "abb" is a follows:



(2)



(3)



- **Deterministic Finite Automata (DFA):**

The finite automata is called deterministic finite automata if there is only one path for a specific input from current state to next state.

Example: The DFA can being shown as below:

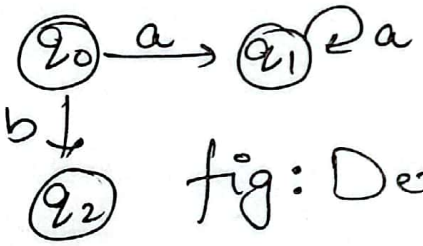


fig: Deterministic finite Automata

- **Definition of DFA:**

A DFA M is described by five tuples $(Q, \Sigma, \delta, q_0, f)$ Q is finite, non-empty set of states. It contains all the states. Σ is an input alphabet. δ is transition func which map $Q \times \Sigma = Q$. q_0 belongs to Q, $q \in Q$ is the initial or starting state. F known as set of final state ($F \subseteq Q$). Simplified Notations used in finite automata.

- **State Transition Graph:**

We represent a finite automata by describing all five tuples $(Q, \Sigma, \delta, q_0, f)$. Transition graph (system) is used for representing finite automata. We used following notations for representing the finite automata.

* The starting state is represented by a state with in a circle and an arrow entering into circle.

→ q_0 (started state named q_0)

* Final state is represented by a state with in double circle.

q_f (final state named q_f)

* The Rang or trap state is represented by the symbol 'ϕ' within a circle. (ϕ)

* Other states are represented by the state name within a circle.

q_1

* A directed edge or arch with label shows the transition (or move)

▲ Suppose p is the present state and q is next state on input symbol 'a' then this is represented by

$$p \xrightarrow{a} q \text{ or } [\delta(p, a) = q]$$

Transition Function:

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

Transition function

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

Direct Transition function :

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

Indirect Transition function:

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

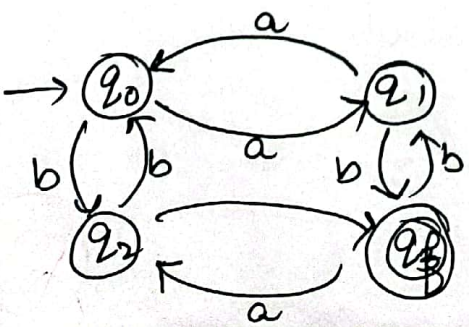
Example : $w = aab$

$$\begin{aligned}\delta(q_0,aab) &= \delta(\delta(q_0,a),ab)\\ &= \delta(q_1,ab)\\ &= \delta(\delta(q_0,a),b)\\ &= \delta(q_2,b)\\ &= q_3.\end{aligned}$$

Transition Table:

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

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



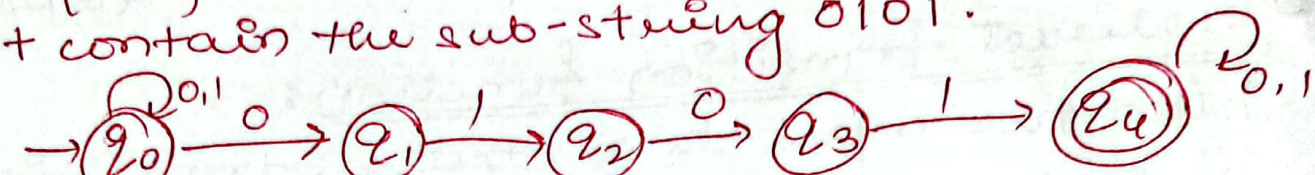
$\delta \mid \Sigma$	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_0	q_f
q_2	q_f	q_0
$*q_f$	q_2	q_1

ted by a st

Language of NFA:

Ques Design a NFA for the lang. L over alphabet $\Sigma = \{0,1\}$ such that every string in the L must contain the sub-string 0101.

Sol^n



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

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

$q_0 = \{q_0\}$ initial State

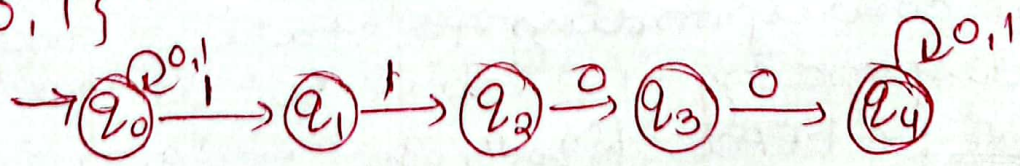
$F = \{q_4\}$ Final state

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

δ / Σ	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	—	$\{q_2\}$
q_2	q_3	—
q_3	—	q_4
$* q_4$	q_4	q_4

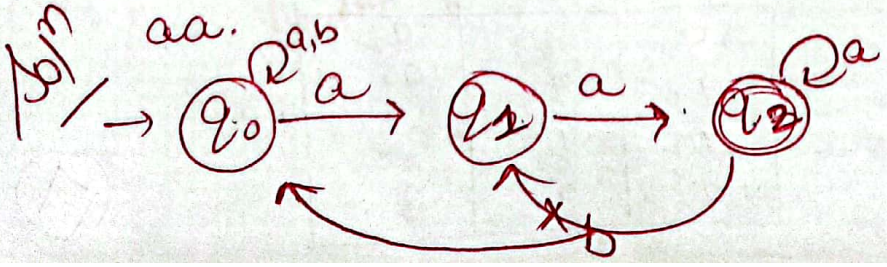
Ques Construct a NFA in which double one is followed by double zero over alphabet $\Sigma = \{0,1\}$

Sol^n



Ques Design a NPA to accept strings with a's and b's such that the string end with aa.

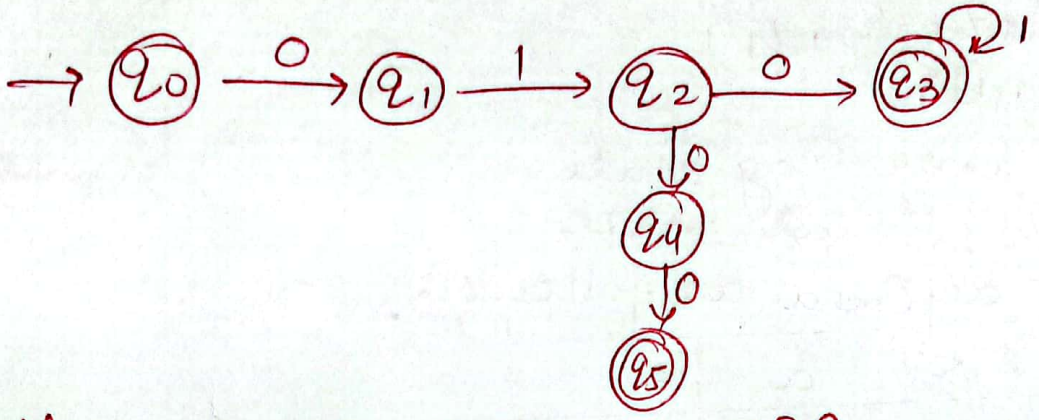
Sol^n



Ques construct NFA for the lang.

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

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

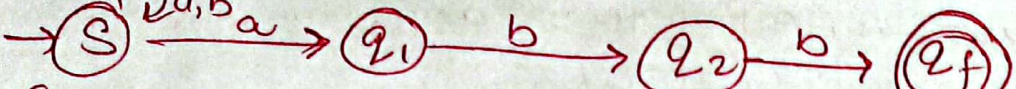


Difference b/w DFA and NFA

S.No	NFA	DFA	INDEX
01.	Same	Same	Power
02.	$Q \times \Sigma \to 2^Q$ (states 0, 1 or more)	$Q \times \Sigma \to Q$ (only 1 or 0)	Transition function
03.	The time needed for executing ~~and~~ an input string is generally more as compare to DFA. Due to Non-Deterministic	The time needed for executing an input string is less as compared to NFA	Time Complexity
04.	All DFA are NFA but not NFA are DFA.	All ~~are~~ DFA are NFA	Supermacy

- Construction of equivalent DFA 'M' from given NFA M1.

Ques Consider a NFA shown in fig.



Solution: let given NFA, $M = (Q, \Sigma, \delta, S, F)$
and its equivalent DFA, $M_1 = (Q_1, \Sigma, \delta_1, [S], F_1)$

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

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

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

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

δ is defined as follows:

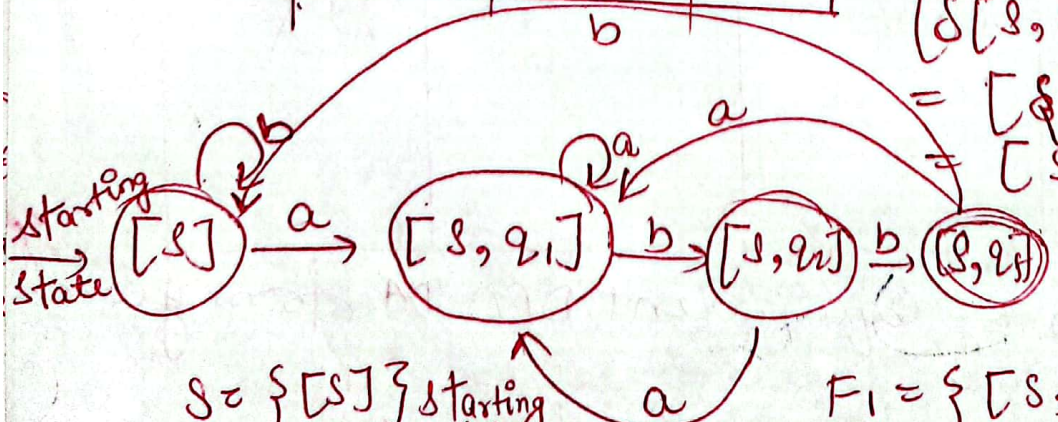
δ / Σ	a	b
$\rightarrow S$	$\{S, q_1\}$	$\{S\}$
q_1	—	$\{q_2\}$
q_2	—	$\{q_f\}$
$* q_f$	—	—

δ_1 is defined as follows:

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

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

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



$S = \{[S]\}$ starting state

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

$F_1 = \{[S, q_f]\}$

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

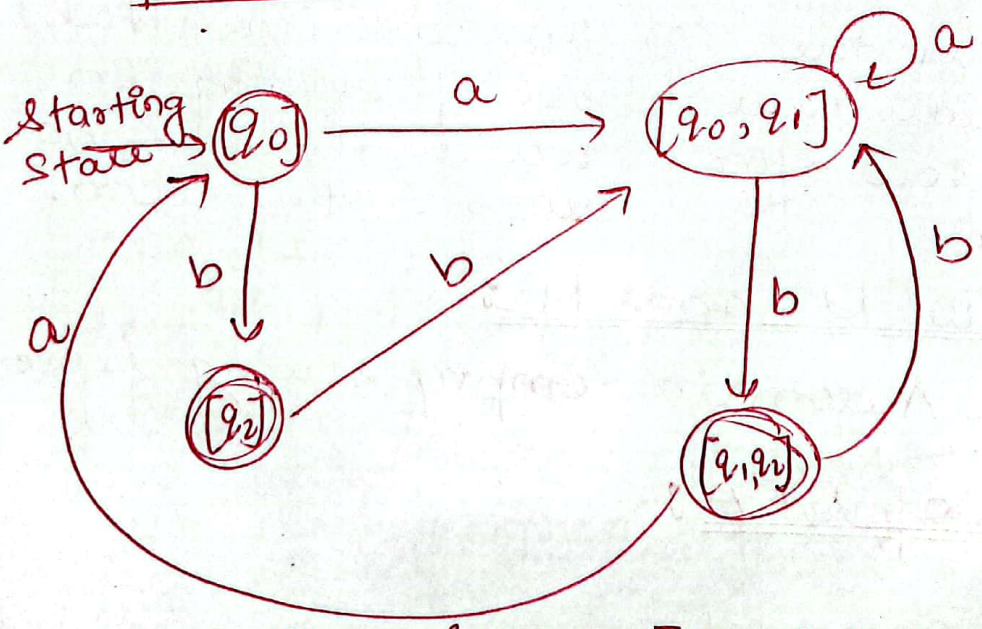
Que Find a DFA equivalent to NFA $M = (\{q_0, q_1, q_2\}, \{a, b\}, \delta, q_0, \{q_2\})$ where δ is defined as follows –

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

Solution :-

Let $M_1 = (Q_1, \Sigma, \delta_1, [q_0], F_1)$ be the equivalent DFA, where $\Sigma = \{a, b\}$, $q_0 = [q_0]$ starting state δ_1 is defined as follows :-

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



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

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

Difference b/w L^* and L^+:

let $\Sigma = \{a\}$.

L^* is a Kleene closure over alphabet $\Sigma = \{a\}$

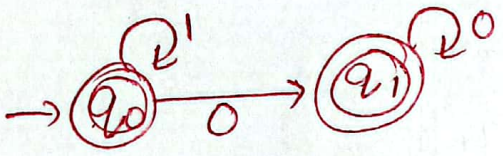
$L^* = \{\varepsilon, a, aa, aaa, aaaa, ---\}$.

L^+ is a positive closure over alphabet $\Sigma = \{a\}$.

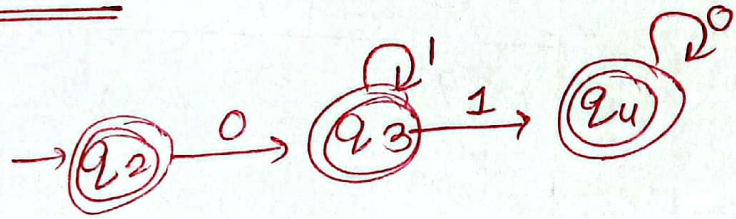
$L^+ = \{a, aa, aaa, aaaa, ---\}$.

Ques show that the Automata M_1 and M_2 is given in figure are equivalent or not. Give the difference and similarity b/w the following FA.

Difference b/w M_1 and M_2:



M_1



M_2

(1) Have two states

(2) $L(M_1)$ will have the string in which there is no one is allow after 0 is encounter

(1) Have three states

(2) $L(M_2)$ will have the set of string in which there is no such restriction.

Similarity b/w M_1 and M_2:

In both the finite Automatas empty string is accepted

Minimization of DFA:

Algorithm:

(1) Eliminate the dead state and unreachable states from the given DFA (if any)

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

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

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

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

④ find the similar rows from T_1 such that:

$\delta(q, a) = P$

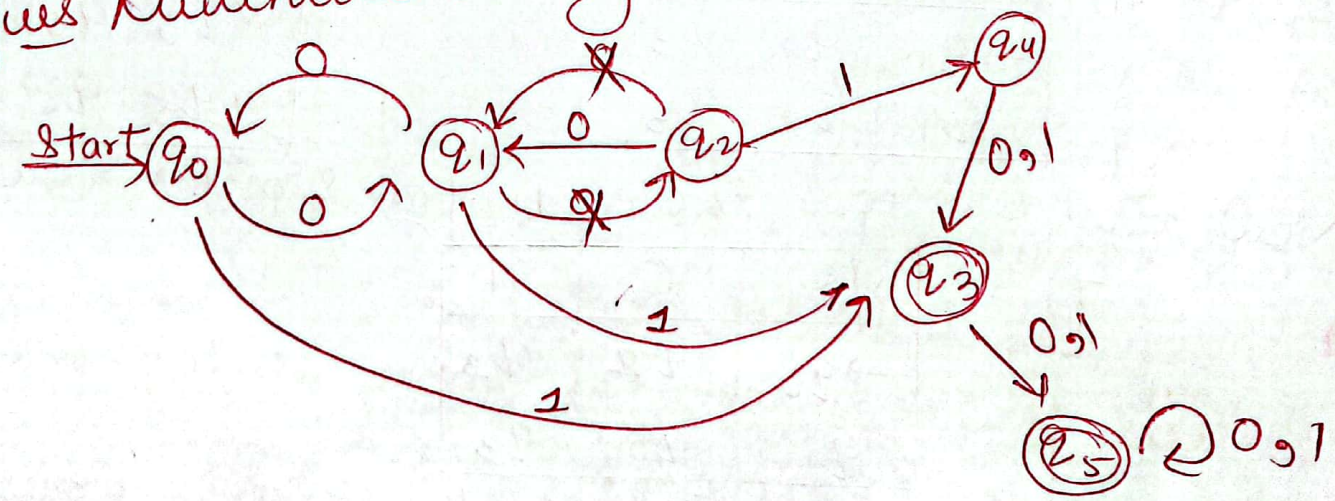
$\delta(q, a) = P$

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

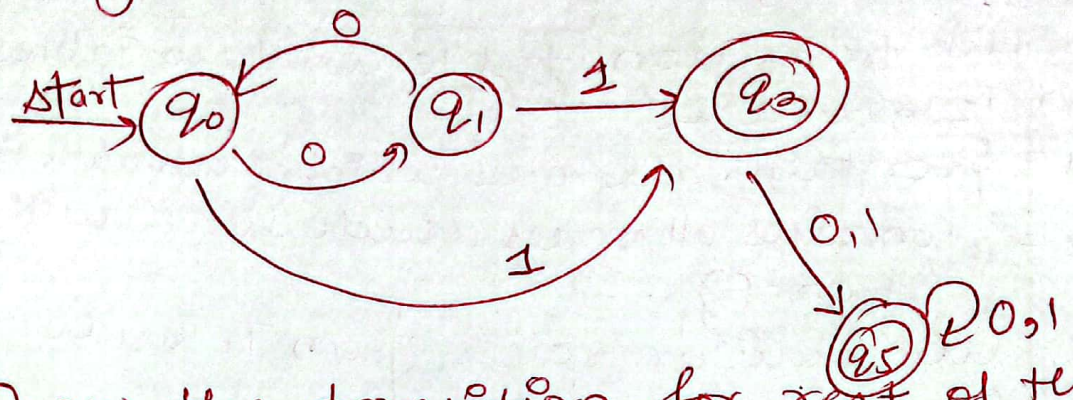
⑥ Repeat the step ④ and step ⑤ for table T_2 also.

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

Ques Minimize the given DFA



(1) Remove the unreachable state q_2 and q_4 from the given DFA.



(2) Draw the transition for rest of the states

δ / Σ	0	1
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_3
$*q_3$	q_5	q_5
$*q_5$	q_5	q_5

(3) Split the transition table into two tables T_1 and T_2.

$T_1 =$

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

$T_2 =$

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

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

$T_1 =$

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

(5) Now consider the table T_2 for finding similar row.

$T_2 =$

δ / Σ	0	1
$*q_3$	$\cancel{q_5}\, q_3$	$\cancel{q_5}\, q_3$
$\cancel{*q_5}$	$\cancel{q_5}$	$\cancel{q_5}$

$T_2 =$

δ / Σ	0	1
$*q_3$	q_3	q_3

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

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

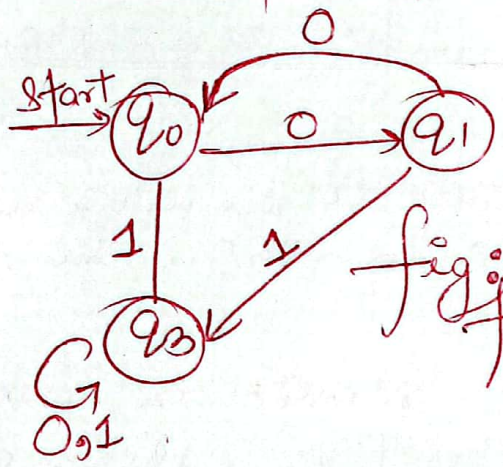


fig: Transition Diagram for minimized DFA.

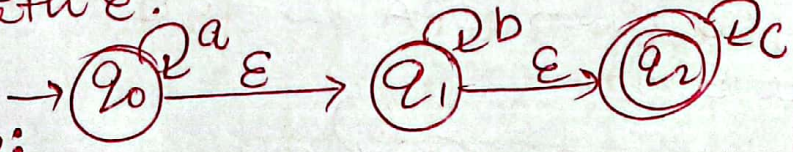
- **Definition of ε-closure:**

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 exist ε closure(p) = {q} $\delta(q, \varepsilon) = r$ then ε closure(p) = {q, r}.

Ques find ε-closure for the following NFA with ε.



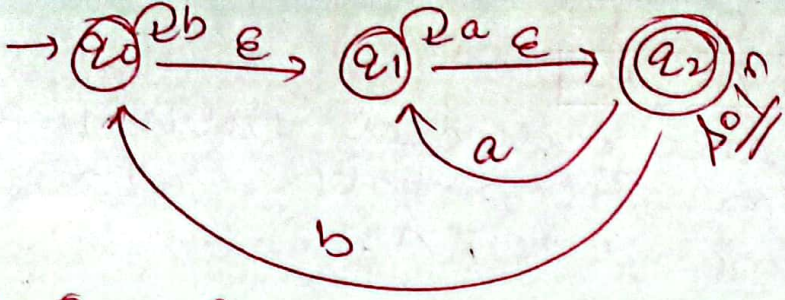
Solution:

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

ε-closure $(q_1) = \{q_1, q_2\}$

ε-closure $(q_2) = \{q_2\}$

Ques



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

Eliminating ε-transition :

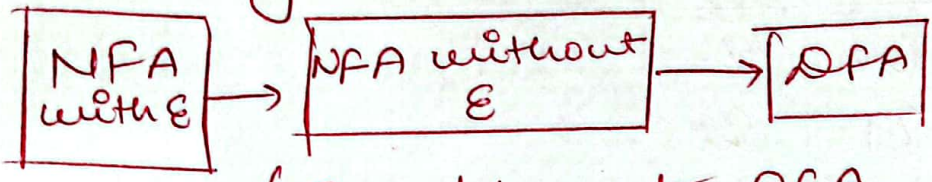


fig: Moves to DFA.

Conversion from NFA with ε to NFA without ε :

In this method we try to remove all the ε transitions from given NFA. The method will be -

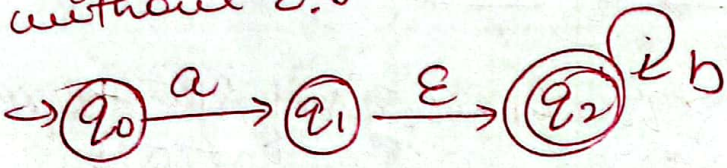
(1) Find out all the ε transitions from each state from Q that will be called as ε-closure $\{q_i\}$, where $q_i \in Q$.

(2) Then δ' transition can be obtained. The δ' transition 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 state the transition table for equivalent NFA without ε can be built.

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



Sol^n ε-closure $(q_0) = \{q_0\}$
ε-closure $(q_1) = \{q_1, q_2\}$
ε-closure $(q_2) = \{q_2\}$.

Now the δ' transitions on each symbol is obtained

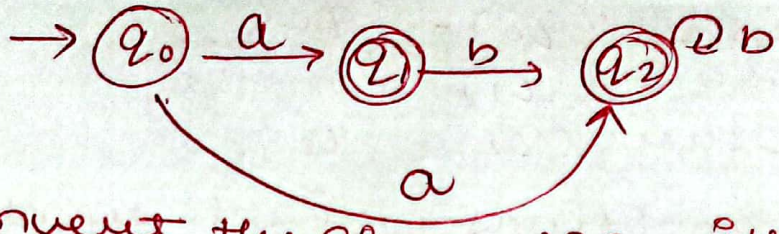
$$\delta'(q_0, a) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_0), a))$$
$$= \varepsilon\text{-closure}(\delta(q_0, a))$$
$$= \varepsilon\text{-closure}(q_1)$$
$$= \{q_1, q_2\}$$

$$\delta'(q_0, b) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_0), b))$$
$$= \varepsilon\text{-closure}(\delta(q_0, b))$$
$$= \varepsilon\text{-closure}(\phi)$$
$$= \phi.$$

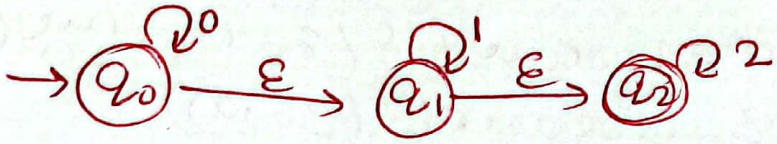
$$\delta'(q_1, a) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_1), a))$$
$$= \varepsilon\text{-closure}(\delta(q_1, a) \cup \delta(q_2, a))$$
$$= \varepsilon\text{-closure}(\phi)$$
$$= \phi$$

$$\delta'(q_1, b) = \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q_1), b))$$
$$= \varepsilon\text{-closure}(\delta(q_1, q_2), b)$$
$$= \varepsilon\text{-closure}(\delta(q_1, b) \cup \delta(q_2, b))$$
$$= \varepsilon\text{-closure}(\phi \cup q_2)$$
$$= \varepsilon\text{-closure}(q_2)$$
$$= \{q_2\}.$$

$\delta \mid \Sigma$	a	b
$\rightarrow q_0$	$\{q_1, q_2\}$	ϕ
$* q_1$	ϕ	$\{q_2\}$
$* q_2$	ϕ	$\{q_2\}$



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



Conversion of NFA with ε to DFA:

(1) Consider $M = (Q, \Sigma, \delta, q_0, F)$ is a NFA with ε. We have to convert this NFA with ε to equivalent DFA denoted by:

$M_D = (Q_D, \Sigma, \delta_D, q_0, F_D)$ then obtain

ε-closure$(q_0) = \{p_1, p_2, p_3 - - p_n\}$ then $[p_1, p_2, p_3 - p_n]$ becomes a start state of DFA

Now $[p_1, p_2, p_3 - - p_n] \in Q_D$

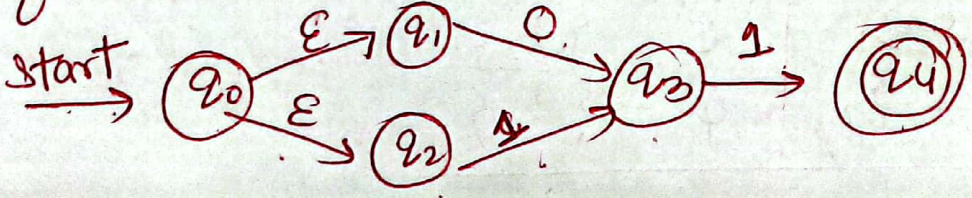
(2) We will obtain δ transition on $[p_1, p_2, p_3 - - p_n]$ for each input $\delta_D([p_1, p_2, p_3 - - p_n], a) =$ ε-closure $(\delta(p_1, a) \cup \delta(p_2, a) \cup \delta(p_3, a) \cup \delta(p_4, a) - - \delta(p_n, a))$

$$\Rightarrow \bigcup_{i=1}^{n} \varepsilon\text{-closure}(p_i, a)$$

where a is input $\in \Sigma$

(3) The states obtained $[p_1, p_2, p_3 - p_n] \in Q_D$ The states containing final state in p_i is a final state in DFA.

Ques Convert the given NFA with ε to its equivalent DFA.



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

ε-closure $(q_1) = \{q_1\}$

ε-closure $(q_2) = \{q_2\}$

ε-closure $(q_3) = \{q_3\}$

ε-closure $(q_4) = \{q_4\}$.

Now, let ε-closure $(q_0) = \{q_0, q_1, q_2\}$ be state A

$$\delta'(A,0) = \varepsilon\text{-closure}(\delta(q_0, q_1, q_2), 0)$$
$$= \varepsilon\text{-closure}(\delta(q_0,0) \cup \delta(q_1,0) \cup \delta(q_2,0))$$
$$= \varepsilon\text{-closure}(\phi \cup q_3 \cup \phi)$$
$$= \varepsilon\text{-closure}(q_3)$$
$$= \{q_3\} \text{ be state } B$$

$$\delta'(A,1) = \varepsilon\text{-closure}(\delta(q_0, q_1, q_2), 1)$$
$$= \quad '' \quad (\delta(q_0,1) \cup \delta(q_1,1) \cup \delta(q_2,1)$$
$$= \quad '' \quad (\phi \cup \phi \cup q_3)$$
$$= \quad '' \quad (q_3)$$
$$= \{q_3\} \text{ be state } B$$

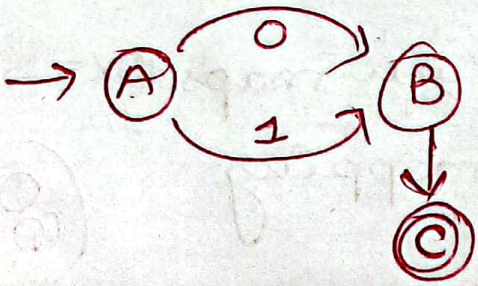
$$\delta'(B,0) = \varepsilon\text{-closure}(\delta(q_3,0))$$
$$= \varepsilon\text{-closure}(\phi)$$
$$= \phi$$

$$\delta'(B,1) = \varepsilon\text{-closure}(\delta(q_3,1))$$
$$= \varepsilon\text{-closure}(q_4)$$
$$= \{q_4\} \text{ be state } C.$$

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

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

δ/Σ	0	1
$\rightarrow A$	B	B
B	ϕ	C
C	ϕ	ϕ



Finite state Machine (FSM) or Transducer:

A finite state m/c is similar to F.A except that it has the additional Capability of producing o/p.

FSM = FA + Output Capability

Types of Finite state Machine:

1. Moore M/c
2. Mealy M/c

Moore Machine:

If the output of FSM is dependent on present state only then this model of FSM is Known as Moore m/c.

Mealy Machine:

If the output of FSM is dependent on present state and present input then this model of FSM is Known as Mealy m/c.

Moore m/c definition:

A Moore m/c can be described by six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$.

(1) Q is a finite set of states.

(2) Σ is the input alphabet.

(3) Δ is the output ~~function~~ alphabet

(4) q_0 is the initial state.

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

(6) λ is the output function mapping $Q \rightarrow \Delta$

Example: Consider the Moore m/c shown in fig. below. Construct the transition table. What is the output for input string 0110?

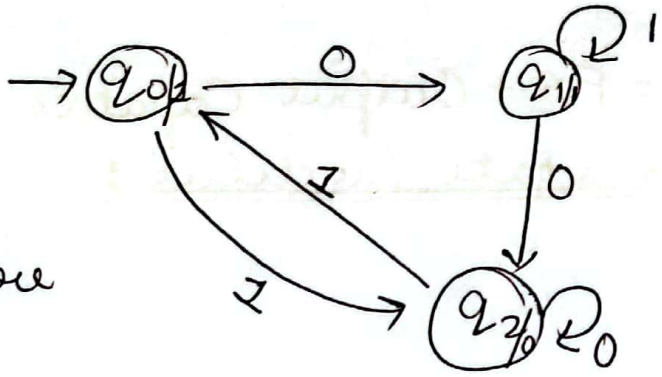
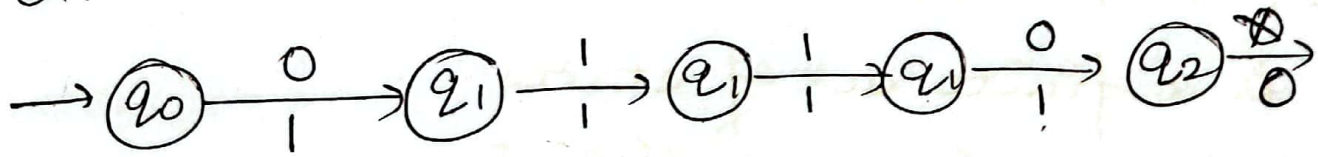


fig: Moore m/c

Sol^n Transition Table

Present state (PS)	Inputs		Outputs
	0	1	
	Next state (NS)	Next state (NS)	*
$\rightarrow q_0$	q_1	q_2	1
q_1	q_2	q_1	1
q_2	q_2	q_0	0

Now the transition sequence for the string 0110 is



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

Note:- For Moore m/c if the input string is of length 'n' the output string is of n+1

Ques What is the o/p for input string 0111 ?

Present
State

Mealy Machine Definition:

A mealy m/c can be described by six tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

(1) Q is finite and non-empty set of states
(2) Σ is input alphabet.
(3) Δ is output alphabet.
(4) δ is transition func which maps $Q \times \Sigma \to Q$
(5) λ is output func. which maps $Q \times \Sigma \to \Delta$
(6) q_0 is the initial state.

Ques Consider the Mealy m/c shown in fig. Construct the transition table & find o/p for Input string 01010?

Soln.

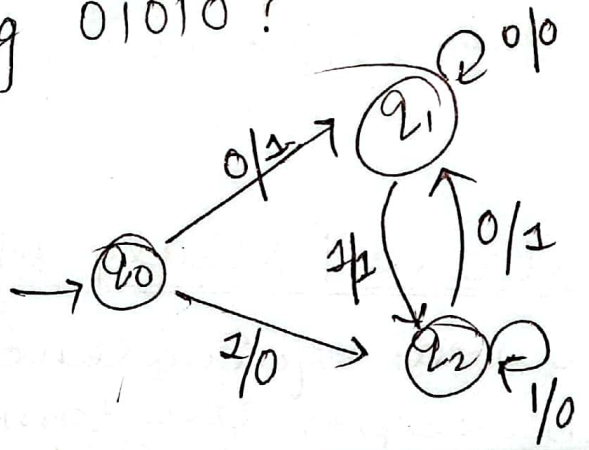


fig: Mealy Machine.

PS	Inputs 0		1	
	NS	O/P	NS	O/P
$\to q_0$	q_1	1	q_2	0
q_1	q_1	0	q_2	1
q_2	q_1	1	q_2	0

Now the transition sequence for the input string 01010

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

O/p → 11111

Ques Find the output for input string 0011?

Soln.

Ps	Inputs 0		Inputs 1	
	Ns	O/P	Ns	O/P
$\rightarrow q_1$	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

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

O/p → 0100.

- **Equivalence of Moore & Mealy M/c:**

The meaning of word equivalence is the two m/cs which accepts the same lang. hence equivalence of Moore & Mealy m/c means both the m/c generate same O/P string or same I/P string.

- **Procedure for transforming a Moore m/c to corresponding to Mealy m/c:**

- **Construction:**

(1) We have to define O/P func. λ' for Mealy m/c as a func of present St

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

to represent any lang. The lang accep

② The transition func is the same as that of the given Moore m/c

Qu Construct a Mealy m/c which is equivalent to the Moore m/c given in transition table.

Sol^n.

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

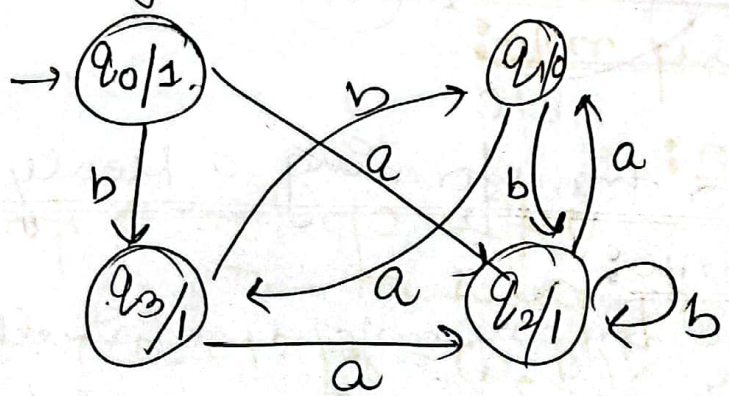
Present state (PS)	Inputs 0 NS	Inputs 0 O/P	Inputs 1 NS	Inputs 1 O/P.
→ q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0

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

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

Qu Change the given Moore m/c into Mealy m/c.

Sol^n.

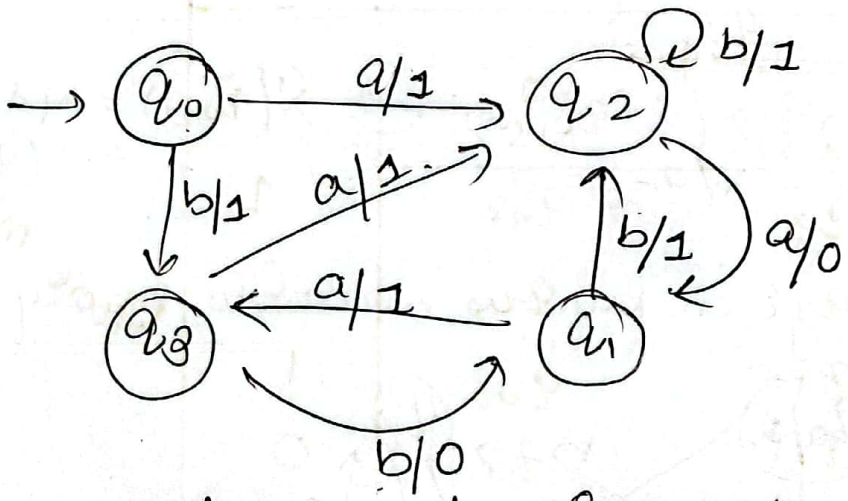


PS	Inputs a NS	Inputs b NS	O/P
$\rightarrow q_0$	q_2	q_3	1
q_1	q_3	q_2	0
q_2	q_1	q_2	1
q_3	q_2	q.	1

fig: Transition for Moore m/c

Present state (PS)	Input a NS	a O/P	b NS	b O/P
$\rightarrow q_0$	q_2	1	q_3	1
q_1	q_3	1	q_2	1
q_2	q_1	0	q_2	1
q_3	q_2	1	q_1	0

fig: Transition Table for Mealy m/c



- **Procedure for transforming a Mealy m/c to a Moore m/c:**

Q. Consider the mealy m/c described by the transition table. Construct the Moore m/c which is equivalent to Mealy m/c.

Present State (PS)	Inputs 0 NS	0 O/P	Inputs 1 NS	1 O/P
→ q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

Mealy m/c

$q_2 < q_{20}, q_{21}$

$q_4 < q_{40}, q_{41}$

Present State (PS)	Inputs 0 NS	0 O/P	Inputs 1 NS	1 O/P
→ q_1	q_3	0	q_{20}	0
q_{20}	q_1	1	q_{40}	0
q_{21}	q_1	1	q_{40}	0
q_3	q_{21}	1	q_1	1
q_{40}	q_{41}	1	q_3	0
q_{41}	q_{41}	1	q_3	0

Mealy m/c

Present State (PS)	Inputs 0 NS	Inputs 1 NS	O/P
→ q_1	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

Moore m/c.

Here we observe that the initial state q_1 is associated without output 1. This means with I/P ε.

we get an output 1. If the m/c start at state q_1. To overcome this situation either we must neglect the response of a Moore m/c to input ε or we must at a new starting state q_0 whose state transitions are identical with those of q_1 with those output is 0.

Present State (PS)	Inputs 0 NS	Inputs 1 NS	Output
$\rightarrow q_0$	q_3	q_{20}	0
q_1	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

Ques What are limitation & Application of FA?

Sol^n **Applications:**

(1) For the designing of lexical analysis of a Compiler.
(2) Used in text editors.
(3) For the implementations of spell checker.
(4) For the designing of transducer.

limitations:

(1) FA can only count finite input.
(2) It can have only string pattern.
(3) Head movement is in only one direction.
(4) Input tape is read only and only m/c it has is state to state.
(5) There is no find and recognize set of binary string of 0's and 1's.
(6) set of strings over "(" and ")" & the have balanced parenthesis.