

Unit: 1

Theory of Computations:-

- ① Symbols :- It could be any symbol that is a to z, A to Z, 0 to 9
- ② Alphabets :- Alphabets are set of symbols (or) collection of symbols. It is represented with ' Σ '. It is finite and non-empty set of symbols.
Eg:- $\Sigma = \{a, b, 0, 1\}$, $\Sigma = \{a-z, A-Z, 0-9\}$
- ③ String :- String is collection of alphabets. It is a finite sequence of symbols chosen from ' Σ '.
Eg :- $\Sigma = \{a, b\}$
a, b, aa, ab, ...
- ④ Language :- Set of strings formed by alphabets (Σ) with some constraints
L1 :- Set of all strings of length 2.
Eg:- $\Sigma = \{a, b\}$ with length 2
 $\Sigma = \{aa, ab, ba, bb\}$
L2 :- Set of all strings where each string starts with 'A'
Eg:- $\Sigma = \{a, b\}$
 $\Sigma = \{a, aa, ab, aab, aba, abb, \dots\}$
- ⑤ Power of alphabets (Sigma ' Σ ') :- power indicates the length of strings over alphabet ' Σ '.
 $\Sigma^0, \Sigma^1 \rightarrow a, b$, $\Sigma^2 \rightarrow ab, ba$, $\Sigma^3 \dots \Sigma^n$ we use notation " Σ^k ".
 $\Sigma^* \Rightarrow \Sigma^0 \cup \Sigma^1 \cup \dots \cup \Sigma^n$
- ⑥ Power of plus :- It represents one or more repetitions of a element.
 $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \dots \cup \Sigma^n$

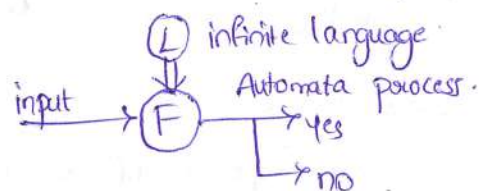
Introduction to finite Automata:-

Finite Automata:-

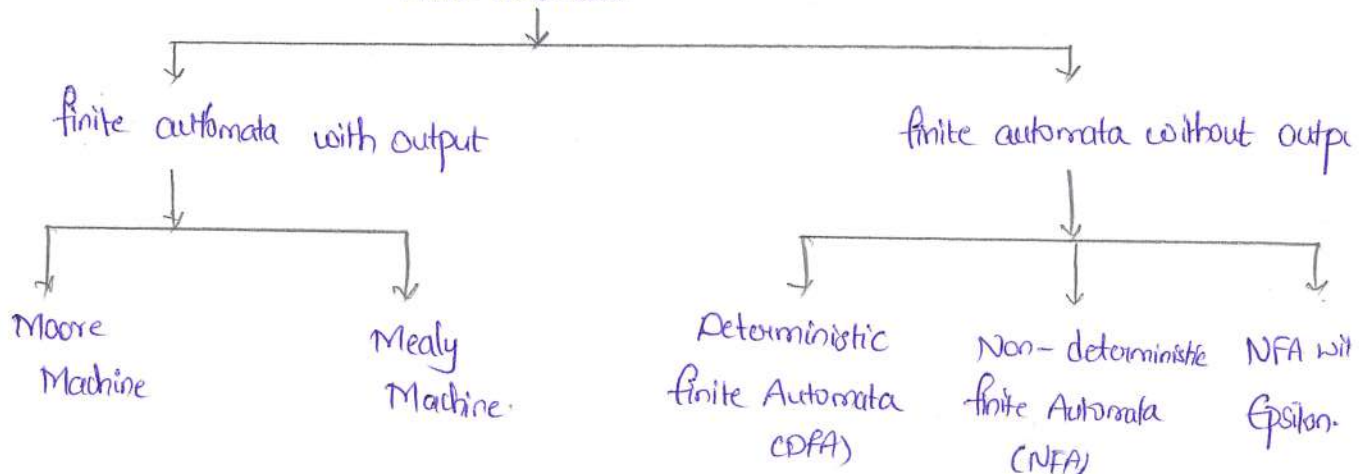
- Converting our representation of infinite strings to finite strings is called as finite automata and giving input as a string and it will validate either it is true or false.
- It is a part of automata theory.
- It is a study of abstract machine.
- Abstract machine are not implemented but represented using some formal notations.
- It is a mathematical model of machine.
- It is behavioural model of machine.

Eg:- If we want to develop a machine which accepts binary number which end with zero can we do it yes $\rightarrow$ 10110 - constraint.

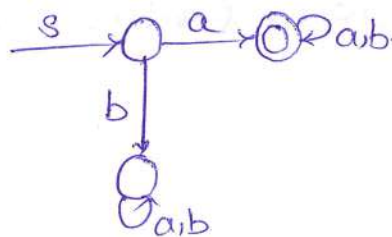
- Can we make a machine who take high level language as input and convert it to binary (yes $\rightarrow$ compiler).
- Can we make a machine who take high level language as input and check that it will never enter into finite loop.
(No the second condition is not possible)



Finite Automata.



Eg- baba by L2 Language.



→○ = Starting point
 ⊙ = Ending point.

String baba is not accepted.

Finite state Machine :-

The simplest module of computation whose memory is very low and performs basic level computation.

Structural representation of finite Automata:-

- Finite state machine consists of a finite set of states and a set of transitions from state to state that occur on input symbols chosen from an alphabet Σ .
- For each input symbol there is exactly one transition out of each state (possibly back to the same state also).
- A directed graph called transition diagram, is associated with an finite automata as follows.
- The finite automata accepts a string x if the sequence of the transitions corresponding to the symbols of x lead from the start state to an accepting state.

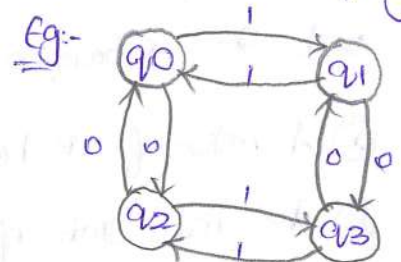
Description of a finite automata:-

A finite automata is represented by 5 tuples.

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

Where

$Q \rightarrow$ is a finite non-empty set of states.

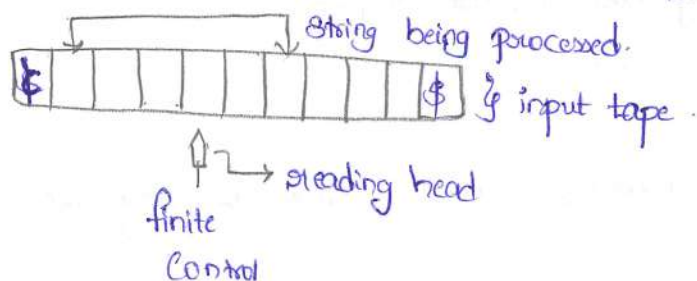


$\Sigma \rightarrow$ is a finite non-empty set of input called input alphabets.

$\delta \rightarrow$ is a function which maps $Q \times \Sigma$ in Q and is usually called direct transition function.

$q_0 \rightarrow$ is initial state ($q_0 \in Q$)

$F \rightarrow$ is the set of final states (can be more than one final state).



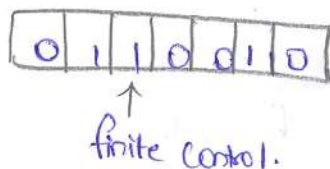
Input tape :- The input tape is divided into squares. Each square containing a single symbol from the input alphabet Σ . The end squares of the tape contain end markers $\$$ at the left end $\$$ at the right end.

- Absence of end markers indicate that the tape is infinite length.
- The left to right sequence of symbols between end markers is the input string to be processed.

Reading Head:- The head examines only one square at a time and can move one square either to the left or to the right. We restrict the movement of R-head only to the right side.

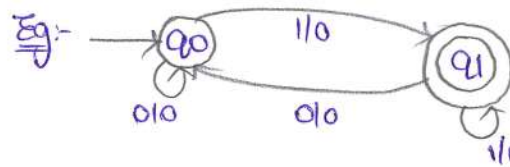
Finite Control:- The input to the finite control will be usually: symbol under the R-head or the present state of the machine.

- A motion of R-head along the tape to the next square.
- The next state of the finite state machine given $\delta(q, a)$



Transition systems:-

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



$1/0 \rightarrow \text{input/output}$

$0/0 \rightarrow \text{input/output}$

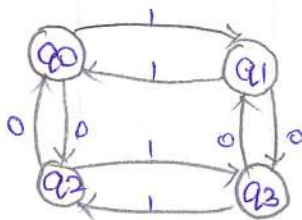
Consider the transition diagram. in our formal notation this FA is denoted as

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

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

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

$F = \{q_3\}$ and δ is shown.



State \ p	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

Sol Suppose 110101 is input to M . we note that $\delta(q_0, 1) = q_1$ and $\delta(q_1, 1) = q_0$

thus $\delta(q_0, 11) = \delta(\delta(q_0, 1), 1)$

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

$$= q_0$$

$\delta(q_0, 0) = q_2$ thus

$$\delta(q_0, 110) = \delta(\delta(q_0, 11), 0)$$

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

$$= q_2$$

$$\delta(q_0, 1101)$$

$$\delta(\delta(q_0, 110), 1)$$

$$\delta(q_3, 1)$$

$$= q_3$$

$$\delta(q_0, 11010)$$

$$\delta(\delta(q_0, 1101), 0)$$

$$\delta(q_3, 0)$$

$$= q_1$$

$$\delta(q_0, 110101)$$

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

$$\delta(q_1, 1)$$

$$= q_0$$

$$\therefore \delta(q_0, 110101) = q_0$$

$\therefore$ The entire sequence of states is



$\therefore 110101$ is in $L(M)$

Automata & Complexity:-

→ Automata are essential for the study of the limits of computation there are two important issues.

1. What can a computer do at all? This study is called "decidability". and the problems that can be solved by computer are called "decidable".
2. What can a computer do efficiently? This study is called "tractability" and the problems that can be solved by a computer using no more time than some slowly growing function of the size of the input are called "tractable".

Length of a string:-

- It is the number of meaningful symbols / alphabets (non- ϵ) present in a string.
- length of a string w , denoted by " $|w|$ "

Eg- if $x = 010100$ then $|x| = 6$.

- if $|w| = 0$ is called an empty string denoted by λ or ϵ

Prefix and Suffix of a string:-

- A prefix of a string is any no of leading symbols of that string.
- A suffix is any no of trailing symbols of that string Example:- String abc

Prefix $\rightarrow \epsilon, a, ab, abc$

Suffix $\rightarrow \epsilon, c, bc, abc$

Concatenation of strings:-

Concatenation of two strings is writing the first followed by the second string with no intervening space.

Eg- ① let x, y be two strings over Σ^* Concatenation of x, y results new string

$$z = xy$$

② let $x = ab, y = pq$ find xy, yx

$$xy = abpq \quad yx = pqab$$

Language and Operations on Languages.

- A Language L over the alphabet Σ is a subset of Σ^* . That is a language is a set of string over the given alphabet.

Operations:-

Union:- It is the simplest operation of two languages. If L_1 and L_2 are two languages then the union denoted $L_1 \cup L_2$ is a language containing all strings w from both languages.

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

Eg $L_1 = \{0, 01, 10, 111\}$ and $L_2 = \{0, 01, 001\}$

$$L_1 \cup L_2 = \{0, 01, 001, 10, 111\}$$

Concatenation:- The Concatenation of languages L_1 and L_2 is defined by

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

Eg ① $L_1 = \{0, 01, 10, 111\}$ $L_2 = \{0, 01, 001\}$

$$L_1 \cdot L_2 = \{00, 001, 0001, 010, 0101, 01001, 100, 1001, 10001, 1110, 11101, 11100\}$$

② Let alphabet $\{0, 1\}$ and L_1, L_2 are the languages over $L_1 = \{0, 01, 1\}$ & $L_2 = \{0, 1\}$

$$L_1 \cdot L_2 = \{00, 01, 00, 01, 110, 111\}$$

$$L_2 \cdot L_1 = \{00, 00, 011, 10, 10, 111\}$$

$$\therefore L_1 \cdot L_2 \neq L_2 \cdot L_1 //$$

The Concatenation operation is however associative.

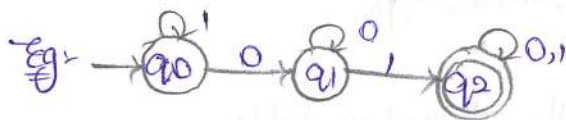
$$\boxed{A(BC) = (AB)C}$$

Problems:-

- In automata theory, a problem is the question of deciding whether a given string is a member of some particular language.
- Anything we more colloquially call a "problem" can be expressed as membership in a language.
- If Σ is an alphabet and L is a language over Σ then the problem L is: Given a string w in Σ^* , decide whether or not w is in L .

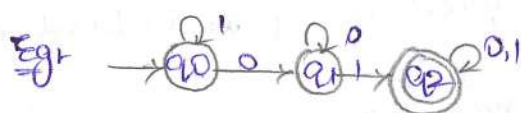
Transition Diagrams and Language Recognizers:-

- It is a directed graph associated with a FA.
- The vertices of the graph correspond to the states of FA.
- If there is a transition from state q to state p on input a , then there is an arc labelled a from state q to state p in the transition diagram.
- The FA accepts a string x if the sequence of transitions corresponding to the symbols of x leads from the start state to an accepting state.
- For each state in Q there is a node in transition diagram.
- For each state q in Q and each input symbol a in Σ if $\delta(q, a) = p$, then there is an arc from node q to node p labelled with a .
- An arrow into the starting state q_0 , labelled start.
- Nodes corresponding to accepting states are marked by double circle.



Transition Tables:-

- A Transition table is a conventional/tabular representation of transition function δ that takes two arguments and returns a value.
- The rows of the table correspond to the states and columns corresponds to the inputs.
- The rows of the table corresponding to the state 'q' and column corresponds to input 'a' is the state $(\delta(q, a))$



State \ ip	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
(q_2)	q_2	q_2

Extending the transition function to strings

- The Extended transition function is a function that takes a state q and a string w and returns a state p .
- It is denoted as $\hat{\delta}$, is a function mapping from $Q \times \Sigma^*$ to Q

$$\hat{\delta}: Q \times \Sigma^* \rightarrow Q$$

$$\text{Let } w = xa$$

$$\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a) = p.$$

Example Construction of $\hat{\delta}$ function the transition table.

Q1 Suppose the input $w = 101101$

Since the string has even no of 0's and it's both are expected it is in the language

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

$$\hat{\delta}(q_0, 101101) = q_0 \text{ (final state)}$$

$$\begin{aligned} \hat{\delta}(q_0, 101101) &= \delta(\delta(q_0, 1), 01101) \\ &= \delta(q_1, 01101) \\ &= \delta(\delta(q_1, 0), 1101) \\ &= \delta(q_3, 1101) \\ &= \delta(\delta(q_3, 1), 101) \\ &= \delta(q_2, 101) \\ &= \delta(\delta(q_2, 1), 01) \\ &= \delta(q_3, 01) \\ &= \delta(\delta(q_3, 0), 1) \\ &= \delta(q_1, 1) \\ &= q_0 \end{aligned}$$

State/Ip	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

i.e string $w = 101101$ is acceptable state.

Properties of transition function:-

Property 1:- $(q, \epsilon) = q$ in a finite automaton. This means the state of the system can be changed only by an input symbol.

Property 2:- for all string w and input symbol a ,

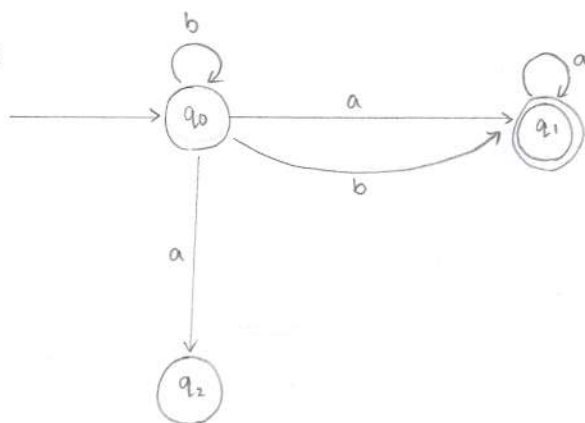
$$\begin{aligned} \delta(q, aw) &= \delta(\delta(q, a), w) \\ \delta(q, wa) &= \delta(\delta(q, w), a) \end{aligned}$$

This property gives that in FA symbol of a string will be processed and then remaining suffix will be processed.

Non-Deterministic Finite Automata (NFA)

- The Finite Automata are called NFA, when there exist many paths for a specific input from the current state to the next state.
- It is easy to construct an 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 the following two exceptions, it contains multiple next states, it contains ϵ transition.

For example:



In the above diagram, we can see that from state q_0 for input as a , there are two next states q_1 and q_2 . Similarly, from q_0 for input b , the next states are q_0 and q_1 . Thus, it is not fixed or determined that with a particular input where to go next. Hence this FA is called non-deterministic finite automata.

Formal Definition of NFA:

NFA also has five states same as DFA, but with different transition functions:

$$\text{i.e. } \delta: Q \times \Sigma \rightarrow 2^Q$$

The five tuples in an NFA are $(Q, \Sigma, \delta, q_0, F)$

Q : finite non-empty set of states

Σ : finite non-empty set of symbols/inputs

q_0 : initial state

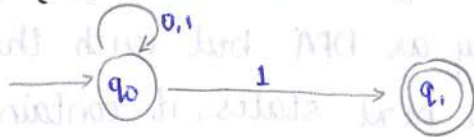
F: set of final states

δ : transition function maps, $\delta: Q \times \Sigma \rightarrow 2^Q$

Examples for NFA

1) Draw an NFA which accepts the language.

$L = \{w/w \text{ ends with } 1\}$



Here

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

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

$$q_0 = q_0$$

$$F = q_1$$

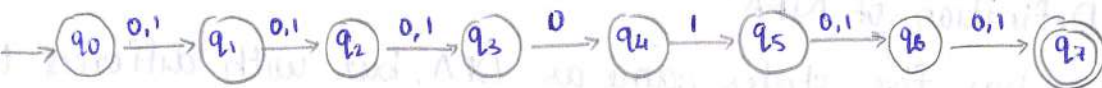
$$\delta(q_0, 0) = \{q_0\}$$

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

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

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

2) Draw NFA which accepts string exactly with 7 characters where the fourth char should always be '0' and fifth should always be '1'



Here

$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$$

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

$$q_0 = q_0$$

$$F = q_7$$

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

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

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

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

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

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

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

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

$$\delta(q_4, 0) = \phi$$

$$\delta(q_4, 1) = q_5$$

$$\delta(q_5, 0) = q_6$$

$$\delta(q_5, 1) = q_6$$

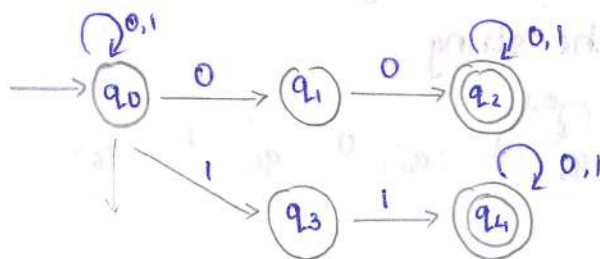
$$\delta(q_6, 0) = q_7$$

$$\delta(q_6, 1) = q_7$$

$$\delta(q_7, 0) = \phi$$

$$\delta(q_7, 1) = \phi$$

3) Draw an NFA that accepts strings with 0's & 1's such that string contains two consecutive 0's or consecutive 1's



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

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

$$q_0 = q_0$$

$$F = \{q_2, q_4\}$$

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

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

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

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

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

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

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

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

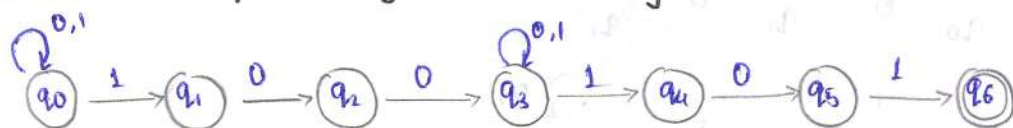
$$\delta(q_4, 0) = \{q_4\}$$

$$\delta(q_4, 1) = \{q_4\}$$

The transition table for the above NFA can be represented as:

i/p	0	1
q_0	q_1, q_0	q_0, q_2
q_1	q_3	ϕ
q_2	q_2	q_2
q_3	ϕ	q_4
q_4	q_4	q_4

4) Draw an NFA which accepts the string that ends with 101 and have 100 somewhere preceding in the string



Here,

$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$$

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

$$q_0 = q_0$$

$$F = q_6$$

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

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

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

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

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

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

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

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

$$\delta(q_4, 0) = \{q_5\}$$

$$\delta(q_4, 1) = \emptyset$$

$$\delta(q_5, 0) = \emptyset$$

$$\delta(q_5, 1) = \{q_6\}$$

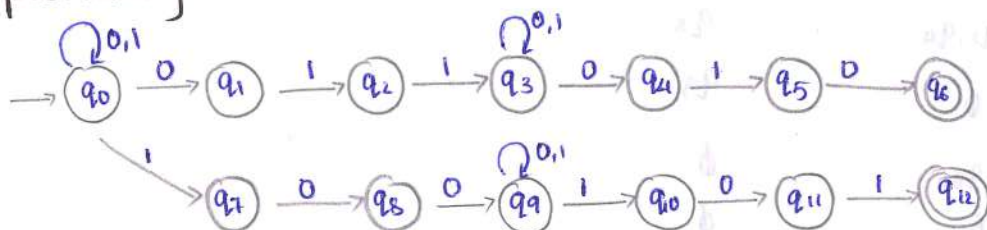
$$\delta(q_6, 0) = \emptyset$$

$$\delta(q_6, 1) = \emptyset$$

The transition table for the above NFA can be represented as

i/p	0	1
q_0	q_0	q_0, q_1
q_1	q_2	$\emptyset$
q_2	q_3	$\emptyset$
q_3	q_3	q_3, q_4
q_4	q_5	$\emptyset$
q_5	$\emptyset$	q_6
q_6	$\emptyset$	$\emptyset$

5) Draw an NFA which accepts the string that ends in 010 and have 011 somewhere preceding or ends in 101 and have 100 somewhere preceding



Here,

$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}\}$$

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

$$q_0 = q_0$$

$$F = \{q_6, q_{12}\}$$

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

$$\delta(q_0, 1) = \{q_0, q_7\}$$

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

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

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

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

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

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

$$\delta(q_4, 0) = \emptyset$$

$$\delta(q_4, 1) = \{q_5\}$$

$$\delta(q_5, 0) = \{q_6\}$$

$$\delta(q_5, 1) = \emptyset$$

$$\delta(q_6, 0) = \emptyset$$

$$\delta(q_6, 1) = \emptyset$$

$$\delta(q_7, 0) = \{q_8\}$$

$$\delta(q_7, 1) = \emptyset$$

$$\delta(q_8, 0) = \{q_9\}$$

$$\delta(q_8, 1) = \emptyset$$

$$\delta(q_9, 0) = \{q_9\}$$

$$\delta(q_9, 1) = \{q_9, q_{10}\}$$

$$\delta(q_{10}, 0) = \{q_{11}\}$$

$$\delta(q_{10}, 1) = \emptyset$$

$$\delta(q_{11}, 0) = \emptyset$$

$$\delta(q_{11}, 1) = \{q_{12}\}$$

$$\delta(q_{12}, 0) = \emptyset$$

$$\delta(q_{12}, 1) = \emptyset$$

The transition table for the above NFA can be represented as :

i/p	0	1
q_0	q_0, q_1	q_0, q_7
q_1	$\emptyset$	q_2
q_2	$\emptyset$	q_3
q_3	q_3, q_4	q_3
q_4	$\emptyset$	q_5
q_5	q_6	$\emptyset$
q_6	$\emptyset$	$\emptyset$
q_7	q_8	$\emptyset$
q_8	q_9	$\emptyset$
q_9	q_9, q_{10}	q_9, q_{10}
q_{10}	$\emptyset, q_{11}$	$\emptyset$
q_{11}	$\emptyset$	q_{12}
q_{12}	$\emptyset$	$\emptyset$

6) Draw an NFA for transition table as given below

i/p	0	1
q_0	$q_0 q_1$	$q_2 q_0$
q_1	q_3	$\emptyset$
q_2	$q_2 q_3$	q_3
q_3	q_3	q_3

From the given transition table, the tuples can be taken as :

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

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

$$q_0 = q_0$$

$$F = q_3$$

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

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

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

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

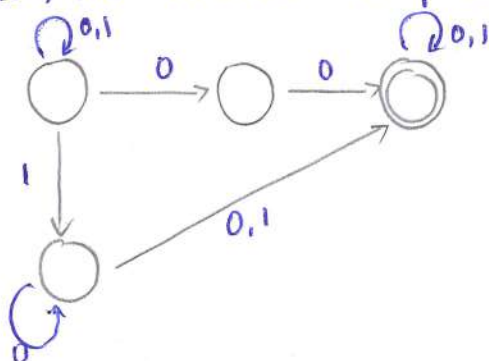
$$\delta(q_2, 0) = \{q_2, q_3\}$$

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

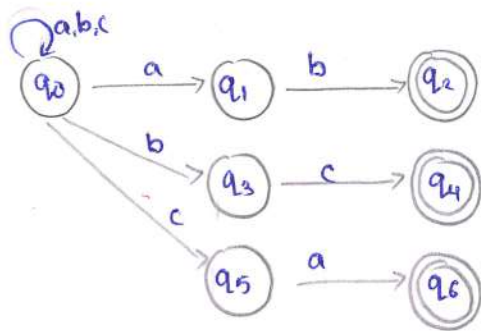
$$\delta(q_3, 0) = \{q_3\}$$

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

Hence, the NFA can be represented as



7) Draw an NFA that accepts the string that ends with one of the following: ab, bc and ca



$$F = \{q_2, q_4, q_6\}$$

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

$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$$

$$q_0 = q_0$$

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

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

$$\delta(q_0, c) = \{q_0, q_5\}$$

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

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

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

$$\delta(q_2, a) = \emptyset$$

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

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

$$\delta(q_3, a) = \emptyset$$

$$\delta(q_3, b) = \emptyset$$

$$\delta(q_3, c) = q_4$$

$$\delta(q_4, a) = \emptyset$$

$$\delta(q_4, b) = \emptyset$$

$$\delta(q_4, c) = \emptyset$$

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

$$\delta(q_5, b) = \emptyset$$

$$\delta(q_5, c) = \emptyset$$

$$\delta(q_6, a) = \emptyset$$

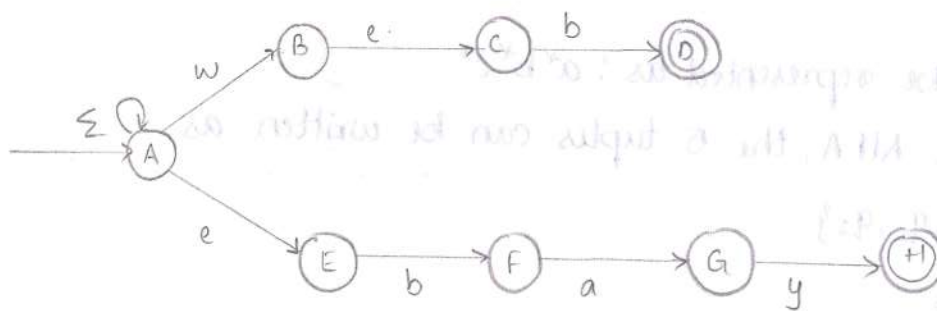
$$\delta(q_6, b) = \emptyset$$

$$\delta(q_6, c) = \emptyset$$

The transition table for the above can be drawn as

q/p	a	b	c
q_0	q_0q_1	q_0q_3	q_0q_5
q_1	$\emptyset$	q_2	$\emptyset$
q_2	$\emptyset$	$\emptyset$	$\emptyset$
q_3	$\emptyset$	$\emptyset$	q_4
q_4	$\emptyset$	$\emptyset$	$\emptyset$
q_5	q_6	$\emptyset$	$\emptyset$
q_6	$\emptyset$	$\emptyset$	$\emptyset$

An application - Text Search.

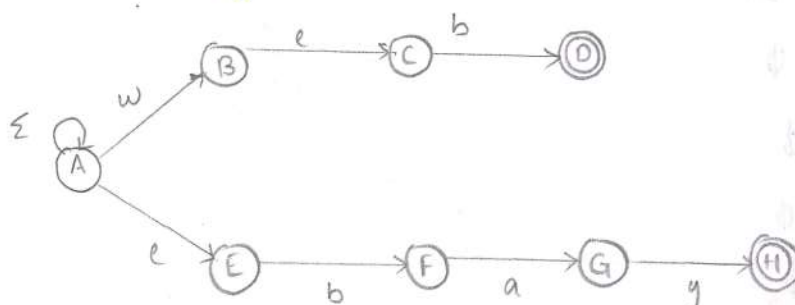


→ Suppose we have a set of words called keywords and we want to find occurrences of any of these words

→ For such problems a useful way to proceed is to design an NFA, which recognizes by entering in an accepting state. There is a sample form to an NFA that recognizes a set of words

Example:

Suppose we want to design an NFA to recognize occurrence of the words web & ebay.



Finite Automato with Epsilon-Transition

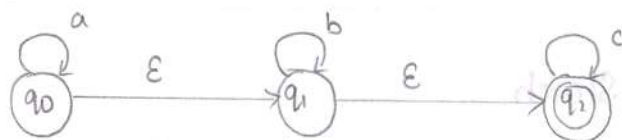
An Epsilon-NFA (ϵ -NFA) is a type of non-deterministic finite automation (NFA) that allows transitions without consuming any input symbols. These transitions are called Epsilon Transitions. This added flexibility makes ϵ -NFAs easier to construct for certain patterns, while still being computationally equivalent to both NFAs and DFAs.

→ ϵ -NFAs add a convenient feature but they bring us nothing new.

They do not extend the class of languages that can be represented.

→ Both NFAs and ϵ -NFAs recognize exactly the same languages.

Write an NFA which accepts a string that indicates 'n' no. of a's, follows with 'n' no. of b's, follows with 'n' no. of c's.



It can be represented as : $a^*b^*c^*$

From the NFA, the 5 tuple can be written as

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

$$q_0 = q_0$$

$$F = F$$

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

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

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

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

$$\delta(q_1, \epsilon) = q_2$$

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

$$\delta(q_2, a) = \phi$$

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

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

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

Draw an NFA for 10^*1^*0



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

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

$$q_0 = q_0$$

$$F = q_3$$

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

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

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

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

$$\delta(q_1, \epsilon) = q_2$$

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

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

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

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

Conversion of NFA with ϵ to NFA without ϵ

Non Deterministic Finite automata (NFA) is a finite automata having zero, one, or more than one moves from a given states on a given input symbol. Epsilon-NFA is the NFA that contains epsilon move(s)/Null moves. To remove the epsilon move/null move from ϵ -NFA and to convert it into NFA, we follow the steps mentioned below.

Step-1: Calculate the ϵ -closure for each state (the set of states reachable from a given state using ϵ -transition)

Step-2: Calculate the extended transition functions (using the following construction formula) and update the transition function

Extended transition function is represented with $\hat{\delta}$

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

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

ϵ -closure of q

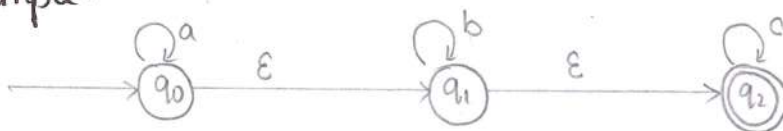
ϵ -closure for a given state x is a set of states which can be reached from the states x with only (null) or ϵ -moves including the state x itself.

The ϵ -closure is as mentioned below-

$$\epsilon\text{-closure}(P) = P, \text{ where } P \in Q$$

• If there exists $\epsilon\text{-closure}(P) = \{q\}$ and $\delta(q, \epsilon) = r$, then $\epsilon\text{-closure}(P) = \{q, r\}$

Example:



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

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

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

$$\begin{aligned} \hat{\delta}(q_0, a) &= \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, \epsilon, a))) \\ &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), a)) \quad [\because \hat{\delta}(q, \epsilon) = \epsilon\text{-closure}(q)] \\ &= \epsilon\text{-closure}(\delta(q_0, q_1, q_2), a) \\ &= \epsilon\text{-closure}(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a)) \\ &= \epsilon\text{-closure}(\delta(q_0, a) \cup \phi \cup \phi) \\ &= \epsilon\text{-closure}(q_0) \\ &= q_0, q_1, q_2 \end{aligned}$$

$$\begin{aligned} \hat{\delta}(q_0, b) &= \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, \epsilon, b))) \\ &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), b)) \quad [\because \hat{\delta}(q, \epsilon) = \epsilon\text{-closure}(q)] \\ &= \epsilon\text{-closure}(\delta(q_0, q_1, q_2), b) \\ &= \epsilon\text{-closure}(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b)) \\ &= \epsilon\text{-closure}(\phi \cup q_1 \cup \phi) \end{aligned}$$

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

$$\hat{\delta}(q_0, c) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, \epsilon, c))) \quad [\because \hat{\delta}(q, \epsilon) = \epsilon\text{-closure}(q)] \\ = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), c)) \\ = \epsilon\text{-closure}(\delta(q_0, q_1, q_2), c) \\ = \epsilon\text{-closure}(\delta(q_0, c) \cup \delta(q_1, c) \cup \delta(q_2, c)) \\ = \epsilon\text{-closure}(\emptyset \cup \emptyset \cup q_2) \\ = q_2$$

$$\hat{\delta}(q_1, a) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1, \epsilon, a))) \\ = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_1), a)) \quad [\because \hat{\delta}(q, \epsilon) = \epsilon\text{-closure}(q)] \\ = \epsilon\text{-closure}(\delta(q_1, q_2), a) \\ = \epsilon\text{-closure}(\delta(q_1, a) \cup \delta(q_2, a)) \\ = \epsilon\text{-closure}(\emptyset \cup \emptyset) \\ = \emptyset$$

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

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

$$\begin{aligned}
 \hat{S}(q_2, a) &= \epsilon\text{-closure}(S(\hat{S}(q_2, \epsilon, a))) \\
 &= \epsilon\text{-closure}(S(\epsilon\text{-closure}(q_2), a)) \\
 &= \epsilon\text{-closure}(S(q_2), a) \\
 &= \epsilon\text{-closure}(S(q_2, a)) \\
 &= \phi
 \end{aligned}$$

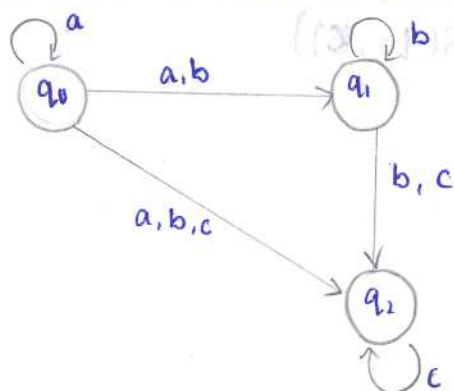
$$\begin{aligned}
 \hat{S}(q_2, b) &= \epsilon\text{-closure}(S(\hat{S}(q_2, \epsilon, b))) \\
 &= \epsilon\text{-closure}(S(\epsilon\text{-closure}(q_2), b)) \\
 &= \epsilon\text{-closure}(S(q_2), b) \\
 &= \epsilon\text{-closure}(S(q_2, b)) \\
 &= \phi
 \end{aligned}$$

$$\begin{aligned}
 \hat{S}(q_2, c) &= \epsilon\text{-closure}(S(\hat{S}(q_2, \epsilon, c))) \\
 &= \epsilon\text{-closure}(S(\epsilon\text{-closure}(q_2), c)) \\
 &= \epsilon\text{-closure}(S(q_2), c) \\
 &= \epsilon\text{-closure}(S(q_2, c)) \\
 &= q_2
 \end{aligned}$$

The transition table for the above can be drawn as

Input Status	a	b	c
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

Hence, the NFA without ϵ can be drawn as



Example - 2

In the above NFA, the 5-tuples are

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

$$q_0 = q_0$$

$$F = q_2$$

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

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

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

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

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

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

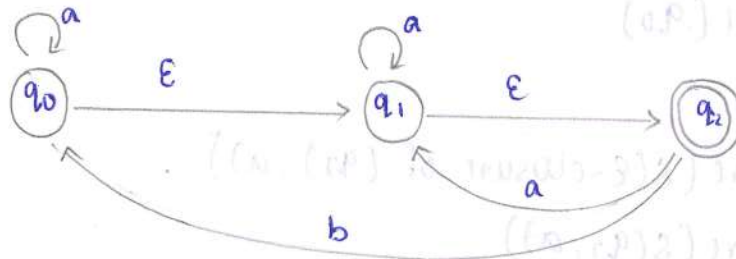
$$\delta(q_1, c) = q_2$$

$$\delta(q_2, a) = \phi$$

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

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

Example - 2



$$\epsilon\text{-closure of } q_0 = q_0 q_1 q_2$$

$$\epsilon\text{-closure of } q_1 = q_1 q_2$$

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

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

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

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

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

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

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

$$\hat{S}(q_0, b) = \epsilon\text{-closure}(S(\epsilon\text{-closure of } (q_0), b))$$

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

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

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

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

$$= q_0 q_1 q_2$$

$$\hat{S}(q_1, a) = \epsilon\text{-closure}(S(\epsilon\text{-closure of } (q_1), a))$$

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

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

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

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

$$= q_1 q_2$$

$$\hat{S}(q_1, b) = \epsilon\text{-closure}(S(\epsilon\text{-closure of } (q_1), b))$$

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

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

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

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

$$= q_0 q_1 q_2$$

$$\hat{S}(q_2, a) = \epsilon\text{-closure}(S(\epsilon\text{-closure of } (q_2), a))$$

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

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

$$= q_1 q_2$$

$$\hat{S}(q_2, b) = \epsilon\text{-closure}(S(\epsilon\text{-closure of } (q_2), b))$$

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

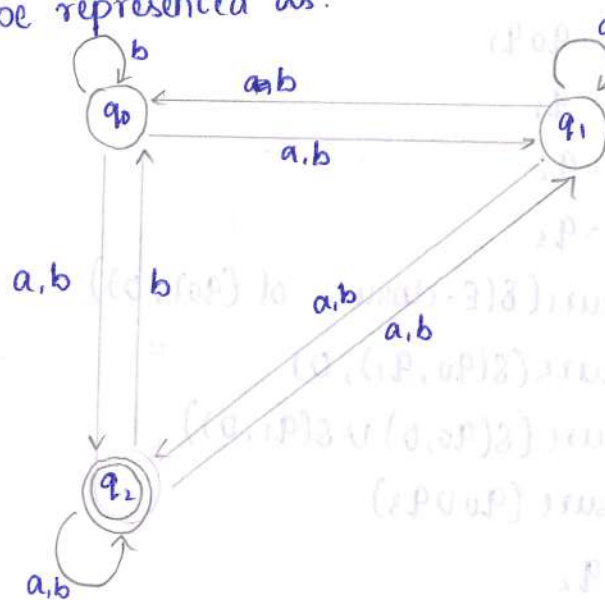
$$= \epsilon\text{-closure}(q_0)$$

$$= q_0 q_1 q_2$$

The transition table can be drawn as :

states \ inputs	a	b
q_0	$q_1 q_2$	$q_0 q_1 q_2$
q_1	$q_1 q_2$	$q_0 q_1 q_2$
q_2	$q_1 q_2$	$q_0 q_1 q_2$

The NFA can be represented as.



The 5-tuples in NFA

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

$$q_0 = q_0$$

$$F = q_2$$

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

$$S(q_0, a) = q_1 q_2$$

$$S(q_0, b) = q_0 q_1 q_2$$

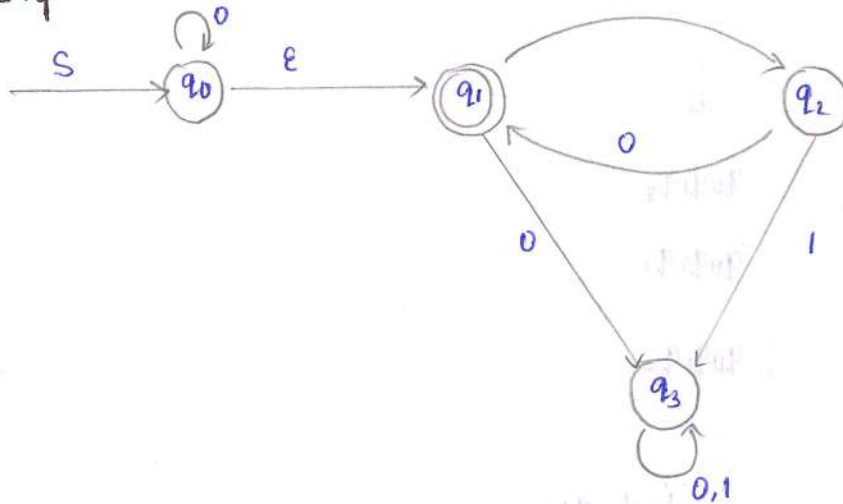
$$S(q_1, a) = q_1 q_2$$

$$S(q_1, b) = q_0 q_1 q_2$$

$$S(q_2, a) = q_1 q_2$$

$$S(q_2, b) = q_0 q_1 q_2$$

Example-3:



ϵ -closure of $q_0 = q_0 q_1$

ϵ -closure of $q_1 = q_1$

ϵ -closure of $q_2 = q_2$

ϵ -closure of $q_3 = q_3$

$$\begin{aligned}
 \hat{S}(q_0, 0) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_0), 0)) \\
 &= \epsilon\text{-closure}(\delta(q_0, q_1), 0) \\
 &= \epsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0)) \\
 &= \epsilon\text{-closure}(q_0 \cup q_3) \\
 &= q_0 q_1 q_2
 \end{aligned}$$

$$\begin{aligned}
 \hat{S}(q_0, 1) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_0), 1)) \\
 &= \epsilon\text{-closure}(\delta(q_0, q_1), 1) \\
 &= \epsilon\text{-closure}(\delta(q_0, 1) \cup \delta(q_1, 1)) \\
 &= \epsilon\text{-closure}(\emptyset \cup q_2) \\
 &= \epsilon\text{-closure}(q_2) \\
 &= q_2
 \end{aligned}$$

$$\begin{aligned}
 \hat{S}(q_1, 0) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_1), 0)) \\
 &= \epsilon\text{-closure}(\delta(q_1), 0) \\
 &= \epsilon\text{-closure}(q_1, 0) \\
 &= \epsilon\text{-closure}(q_3) \\
 &= q_3
 \end{aligned}$$

$$\begin{aligned}
 \hat{S}(q_1, 1) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_1), 1)) \\
 &= \epsilon\text{-closure}(\delta(q_1), 1) \\
 &= \epsilon\text{-closure}(\delta(q_1, 1)) \\
 &= \epsilon\text{-closure}(q_2) \\
 &= q_2
 \end{aligned}$$

$$\begin{aligned}
 \hat{S}(q_2, 0) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_2), 0)) \\
 &= \epsilon\text{-closure}(\delta(q_2), 0) \\
 &= \epsilon\text{-closure}(\delta(q_2, 0)) \\
 &= \epsilon\text{-closure}(q_1) \\
 &= q_1
 \end{aligned}$$

$$\begin{aligned}
 \hat{S}(q_2, 1) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_2), 1)) \\
 &= \epsilon\text{-closure}(\delta(q_2), 1) \\
 &= \epsilon\text{-closure}(\delta(q_2, 1)) \\
 &= \epsilon\text{-closure}(q_3) \\
 &= q_3
 \end{aligned}$$

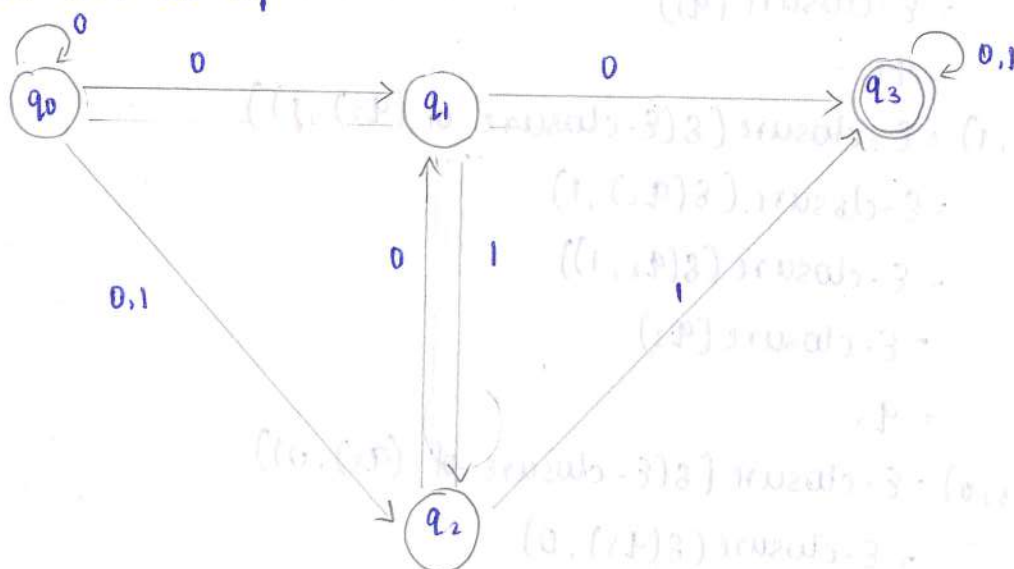
$$\begin{aligned}
 \hat{S}(q_3, 0) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_3), 0)) \\
 &= \epsilon\text{-closure}(\delta(q_3), 0) \\
 &= \epsilon\text{-closure}(\delta(q_3, 0)) \\
 &= \epsilon\text{-closure}(q_3) \\
 &= q_3
 \end{aligned}$$

$$\begin{aligned}
 \hat{S}(q_3, 1) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_3), 1)) \\
 &= \epsilon\text{-closure}(\delta(q_3), 1) \\
 &= \epsilon\text{-closure}(\delta(q_3, 1)) \\
 &= \epsilon\text{-closure}(q_3) \\
 &= q_3
 \end{aligned}$$

The transition table for this can be drawn as.

inputs status	0	1
q_0	$q_0 q_1 q_2$	q_2
q_1	q_3	q_2
q_2	q_1	q_3
q_3	q_3	q_3

The NFA can be represented as



The 5-tuples in NFA

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

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

$$q_0 = q_0$$

$$F = q_3$$

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

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

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

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

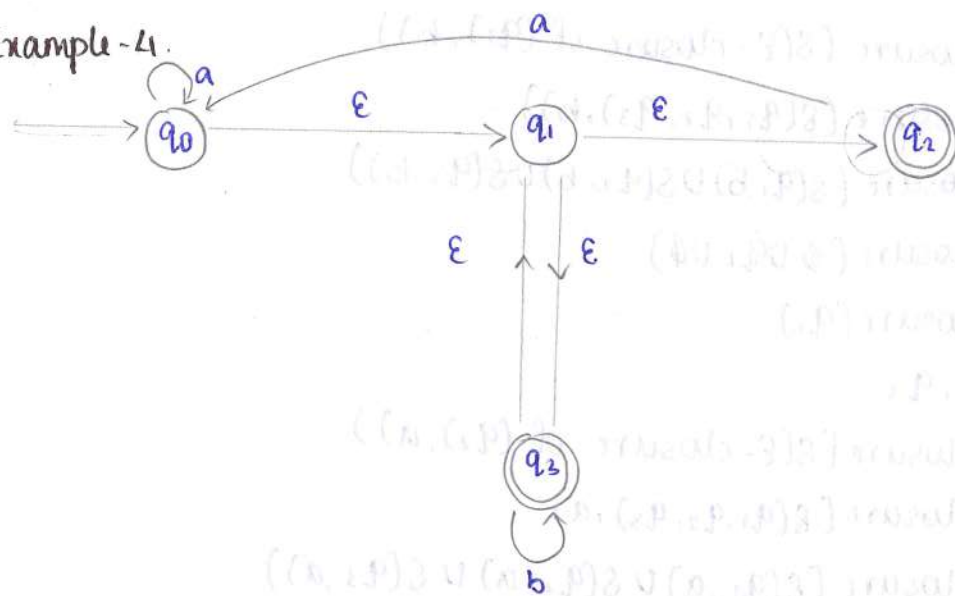
$$\delta(q_2, 0) = q_1$$

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

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

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

Example - 4.



ϵ -closure of $q_0 = q_0, q_1, q_2, q_3$

ϵ -closure of $q_1 = q_1, q_2, q_3$

ϵ -closure of $q_2 = q_2, q_1, q_3$

ϵ -closure of $q_3 = q_3$

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

$= \epsilon\text{-closure}(\delta(q_0, q_1, q_2, q_3), a)$

$= \epsilon\text{-closure}(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a))$

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

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

$= q_0, q_1, q_2, q_3$

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

$= \epsilon\text{-closure}(\delta(q_0, q_1, q_2, q_3), b)$

$= \epsilon\text{-closure}(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b))$

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

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

$= q_1, q_2, q_3$

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

$= \epsilon\text{-closure}(\delta(q_1, q_2, q_3), a)$

$= \epsilon\text{-closure}(\delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a))$

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

$= \epsilon\text{-closure}(q_0) = q_0, q_1, q_2, q_3$

$$\begin{aligned}
 \hat{\delta}(q_1, b) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_1), b)) \\
 &= \epsilon\text{-closure}(\delta(q_1, q_2, q_3), b) \\
 &= \epsilon\text{-closure}(\delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b)) \\
 &= \epsilon\text{-closure}(\emptyset \cup q_2 \cup \emptyset) \\
 &= \epsilon\text{-closure}(q_2) \\
 &= q_1 q_2 q_3
 \end{aligned}$$

$$\begin{aligned}
 \hat{\delta}(q_2, a) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_2), a)) \\
 &= \epsilon\text{-closure}(\delta(q_1, q_2, q_3), a) \\
 &= \epsilon\text{-closure}(\delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a)) \\
 &= \epsilon\text{-closure}(\emptyset \cup \emptyset \cup q_0) \\
 &= q_0 q_1 q_2 q_3
 \end{aligned}$$

$$\begin{aligned}
 \hat{\delta}(q_2, b) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_2), b)) \\
 &= \epsilon\text{-closure}(\delta(q_1, q_2, q_3), b) \\
 &= \epsilon\text{-closure}(\delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b)) \\
 &= \epsilon\text{-closure}(\emptyset \cup q_2 \cup \emptyset) \\
 &= \epsilon\text{-closure}(q_2) \\
 &= q_1, q_2, q_3
 \end{aligned}$$

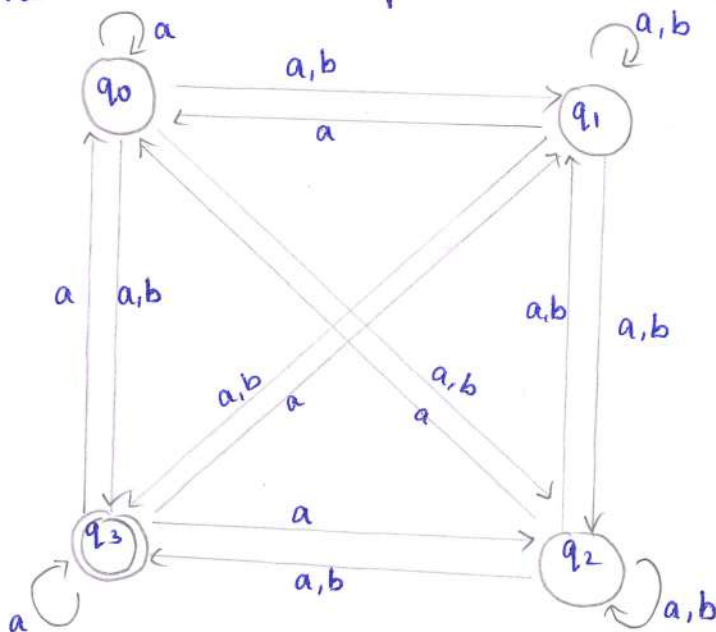
$$\begin{aligned}
 \hat{\delta}(q_3, a) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_3), a)) \\
 &= \epsilon\text{-closure}(\delta(q_3), a) \\
 &= \epsilon\text{-closure}(\delta(q_3, a)) \\
 &= \epsilon\text{-closure}(q_0) \\
 &= q_0 q_1 q_2 q_3
 \end{aligned}$$

$$\begin{aligned}
 \hat{\delta}(q_3, b) &= \epsilon\text{-closure}(\delta(\epsilon\text{-closure of } (q_3), b)) \\
 &= \epsilon\text{-closure}(\delta(q_3), b) \\
 &= \epsilon\text{-closure}(\delta(q_3, b)) \\
 &= \epsilon\text{-closure}(\delta(q_3, b)) \\
 &= \emptyset
 \end{aligned}$$

The transition table of the NFA can be drawn as

inputs status	a	b
q_0	$q_0 q_1 q_2 q_3$	$q_1 q_2 q_3$
q_1	$q_0 q_1 q_2 q_3$	$q_1 q_2 q_3$
q_2	$q_0 q_1 q_2 q_3$	$q_1 q_2 q_3$
q_3	$q_0 q_1 q_2 q_3$	ϕ

The NFA can be represented as:



The 5-tuples of NFA

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

$$q_0 = q_0$$

$$F = q_3$$

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

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

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

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

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

$$S(q_2, a) = q_0 q_1 q_2 q_3$$

$$S(q_2, b) = q_1 q_2 q_3$$

$$S(q_3, a) = q_0 q_1 q_2 q_3$$

$$S(q_3, b) = \phi$$

The DFA can be represented as:



Deterministic Finite Automata.

DFA refers to Deterministic Finite Automata. Deterministic refers to the uniqueness of the computation. The finite automata are deterministic FA if the machine reads an input string one symbol at a time.

In DFA, there is only one path input from the current state to the next state. It does not accept the null move, i.e. it cannot change state without any input. It can contain multiple final states.

In DFA, for each input symbol, one can determine the state to which the machine will move. Hence, it is called Deterministic Automation. As it has a finite number of states, the machine is called Deterministic Finite Machine or Deterministic Finite Automation.

Formal Definition of a DFA.

A Deterministic Finite Automata (DFA) is a collection of 5-tuples and they are as follows

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

Where,

Q is a finite set of states

Σ is a finite set of symbols called the alphabet

δ is the transition function where $\delta: Q \times \Sigma \rightarrow Q$

q_0 is the initial state from where any input is processed

F is a set of final state / states of Q .

How a DFA processes string:-

→ DFA starts with initial state q_0 , let $a_1, a_2, \dots, a_n$ is a sequence of input symbols and δ is transition function.

Step 1:- $\delta(q_0, a_1) = q_1$, DFA in state q_0 on input a_1 enters state q_1 .

Step 2:- $\delta(q_1, a_2) = q_2$, DFA in state q_1 on input a_2 enters state q_2 .

Similarly $\delta(q_{i-1}, a_i) = q_i$ for each i

→ if q_n is a member of F , then input $a_1, a_2, \dots, a_n$ is accepted otherwise it will not be accepted by DFA.

Example Consider the finite state machine whose transition δ is given in the transition table Here $Q = \{q_0, q_1, q_2, q_3\}$

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

$$F = \{q_0\}$$

input string = 110101

Transition Table:-

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

Sol, $\delta(q_0, 110101)$

$$\delta(q_1, 10101)$$

$$\delta(q_0, 0101)$$

$$\delta(q_2, 101)$$

$$\delta(q_3, 01)$$

$$\delta(q_1, 1) \rightarrow q_0$$

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

∴ The given string is 110101.

Language of DFA :-

→ The Language of DFA $M = (Q, E, \delta, q_0, F)$ is denoted by $L(M)$ is defined as:

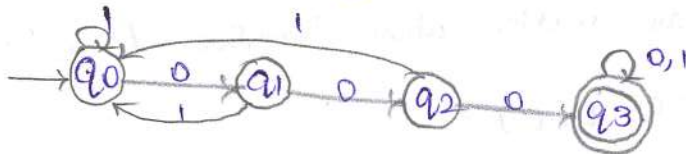
$$L(M) = \{w \mid \delta^*(q_0, w) \text{ is in } F\}$$

→ That is the language of M is the set of strings w that take initial state q_0 to one of the accepting states.

→ If L is $L(M)$ for some DFA M , then L is called a regular language.

Examples:-

① Construct a DFA accepting the set of all strings with three consecutive 0's.

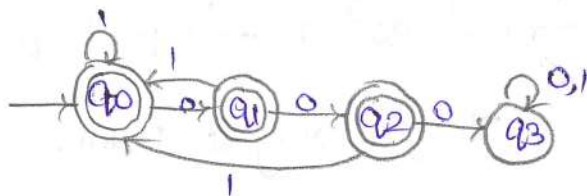


Transition table.

State	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_3	q_0
q_3	q_3	q_3

② Design a DFA M that accepts the language $L(M) = \{w \mid w \in \{0,1\}^* \text{ and } w \text{ doesn't contain 3 consecutive 0's}\}$.

We get DFA that accepts strings without three consecutive 0's by making all non-final states as final states & final states as non-final states.



Transition Table:-

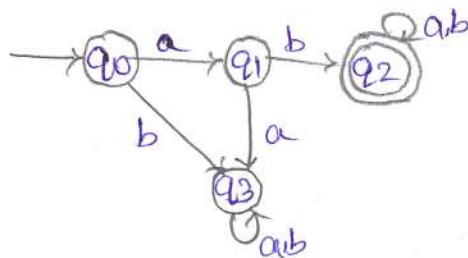
State/p	0	1
q ₀	q ₁	q ₀
q ₁	q ₃	q ₀
q ₂	q ₃	q ₀
q ₃	q ₃	q ₃

Examples:-

Note- All strings starting with 'i' length substring will always requires minimum $(n+2)$ states in the DFA.

① Draw a DFA for the language accepting strings starting with 'ab' over input alphabets $\Sigma = \{a, b\}$

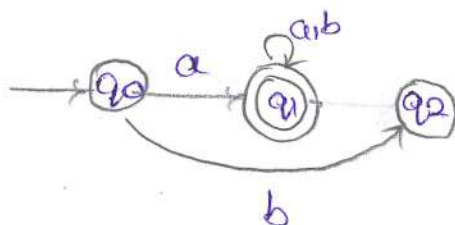
Sol- All strings of the language start with substring "ab". i.e., length of substring = 2
Thus minimum no of states required in the DFA $2+2=4$.



② Draw a DFA for the language accepting strings starting with 'a' over input alphabets $\Sigma = \{a, b\}$

Substring = 1

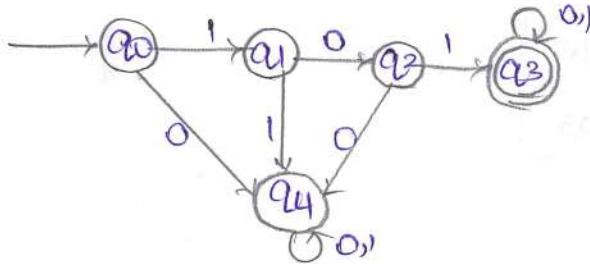
$\therefore$ no of states = $1+2=3$.



3) Draw a DFA for the language accepting strings with '101' over input alphabet $\Sigma = \{0,1\}$

sol Substring = 101 = 3

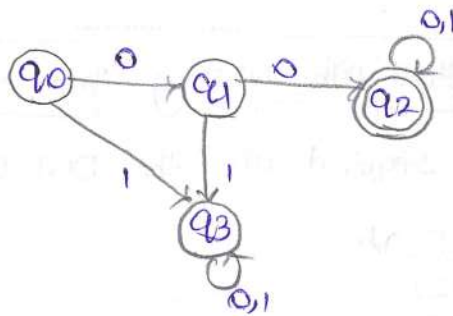
no of nodes = 3+2 = 5.



4) Draw a DFA that accepts a language 'L' over input alphabets $\Sigma = \{0,1\}$ such that L is the set of all strings starting with '00'.

sol Substring = 00 = 2

no of nodes = 2+2 = 4



Equivalence of NFA and DFA

→ FA M_1 and M_2 are said to be equivalent if and only if $L(M_1) = L(M_2)$

→ Two automata are considered to be equivalent if they accept the same language even though they use different methods.

Theorem:-

Let L be a set accepted by a NFA, then there exists a DFA that accepts L .

Proof:- Let $M = \{Q, \Sigma, \delta, q_0, F\}$ be an NFA accepting L .

We can construct DFA " M' " as follows:

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

where

1. $Q' = 2^Q$ (any state in Q' is denoted by $[q_1, q_2, \dots, q_j]$ where $q_1, q_2, \dots, q_j \in Q$)

2. $q_0' = [q_0]$

3. F' is the set of all subsets of Q containing an element of F .

4. We define

$$\delta'([q_1, q_2, \dots, q_j], a) = [q_1, q_2, \dots, q_j]$$

if and only if

$$\delta([q_1, q_2, \dots, q_j], a) = [q_1, q_2, \dots, q_j]$$

Basis:- the result is trivial function $|x|=0$ since $q_0' = [q_0]$ and x must be ϵ .

Induction:- Suppose that the hypothesis is true function input of length m or less.

Let x_a is a string of length $m+1$ with $a \in \Sigma$ then

$$\delta'(q_0', x_a) = \delta'(\delta'(q_0', x), a)$$

Hypothesis:-

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

if and only if

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

But by definition of δ'

$$\delta'([p_1, p_2, \dots, p_j], a) = \{\delta(p_1, a), \delta(p_2, a), \dots, \delta(p_j, a)\}$$

if and only if

$$\delta(p_1, p_2, \dots, p_{j-1}, a) = [x_1, x_2, \dots, x_k]$$

$$\text{thus } \delta'(q_0, x_1 x_2 \dots x_k) = [x_1, x_2, \dots, x_k]$$

if and only if

$$\delta(q_0, x_1) = [x_1, x_2, \dots, x_k]$$

$\therefore \delta'(q_0, x_1)$ is in F Exactly when $\delta(q_0, x_1)$

Contains a state of Q that is in F .

$$\text{Thus } L(m) \subseteq L(m')$$

$\therefore$ There exists a DFA Equivalent to NFA.

Conversion of NFA to DFA

An NFA can have zero, one or more than one move from a given state on a given input symbol. An NFA can also have NULL moves (moves without input symbol). On the other hand, DFA has one and only one move from a given state on a given input symbol.

Steps for Converting NFA to DFA

Step 1: Convert the given NFA to its equivalent transition table

To convert the NFA to its equivalent transition table, we need to list all the states, input symbols, and the transition rules. The transition rules are represented in the form of a table, where

Step 2: Create the DFA's transition table

The DFA's transition table is similar to the NFA's transition table, but instead of individual states, the rows and columns represent set of states.

Step 3: Construct a DFA

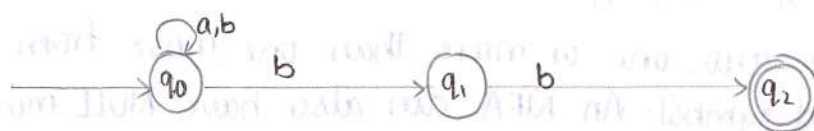
The DFA's start state is the set of all possible starting states in the NFA.

The DFA's final states are the set of states that contain at least one final state from the NFA.

To simplify the DFA, we can use the following techniques:

- Remove dead states: States that cannot lead to final state can be removed.
- Merge equivalent states: States that have the same transition rules of all input symbols can be merged into a single state.

Example-1



Step-1: Write the transition table of NFA

	a	b
q_0	q_0	q_0q_1
q_1	ϕ	q_2
q_2	ϕ	ϕ

Step-2: Convert into DFA transition table

	a	b
q_0	q_0	$\{q_0, q_1\}$
$\{q_0, q_1\}$	q_0	$\{q_0, q_1, q_2\}$
$\{q_0, q_1, q_2\}$	q_0	$\{q_0, q_1, q_2\}$

From the above NFA transition table, we write initial state's transition rules i.e. $\delta(q_0, a) = q_0$, $\delta(q_0, b) = \{q_0, q_1\}$.

Now the next state is the transition rule of initial state.

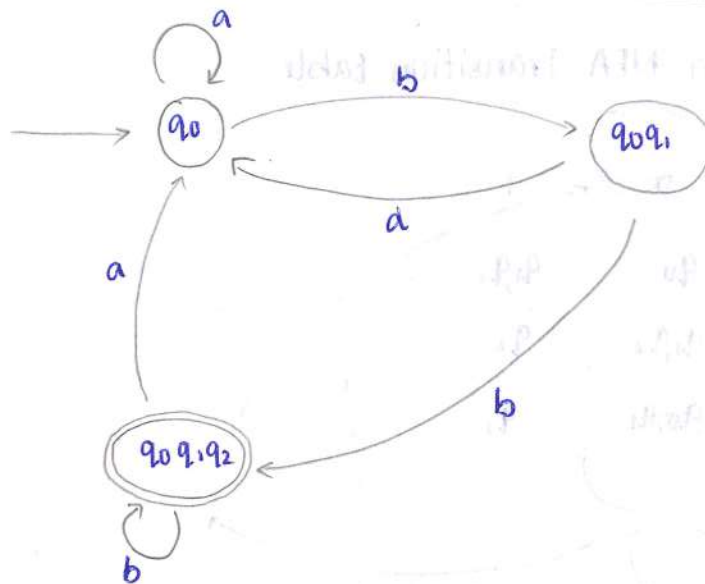
i.e. $\delta(\{q_0, q_1\}, a) = q_0$, $\delta(\{q_0, q_1\}, b) = \{q_0, q_1, q_2\}$

The next state will be the another transition rule of the previous state.

i.e. $\delta(\{q_0, q_1, q_2\}, a) = q_0$, $\delta(\{q_0, q_1, q_2\}, b) = \{q_0, q_1, q_2\}$

Step-3: From The Construction of DFA-

From the above DFA's transition table, we draw a DFA



According to NFA, the final state is q_2 . So in DFA, the final state will be all the states with q_2

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

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

$$q_0 = q_0$$

$$F = q_0q_1q_2$$

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

$$\delta(q_0, b) = \{q_0q_1\}$$

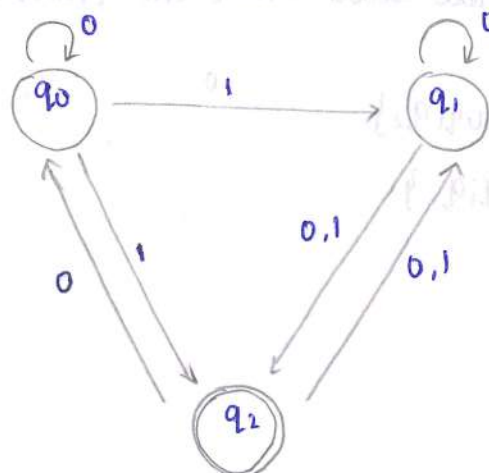
$$\delta(q_0q_1, a) = q_0$$

$$\delta(q_0q_1, b) = \{q_0q_1q_2\}$$

$$\delta(q_0q_1q_2, a) = q_0$$

$$\delta(q_0q_1q_2, b) = \{q_0q_1q_2\}$$

Example - 2.



Step-1) Constructing an NFA transition table.

input states	0	1
q_0	q_0	q_1, q_2
q_1	q_1, q_2	q_2
q_2	q_0, q_1	q_1

Step-2) Converting into DFA transition table.

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

For DFA transition table, the transition rules for the initial state are derived from NFA transition table.

The next state will be the state from the transition rules of initial states

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

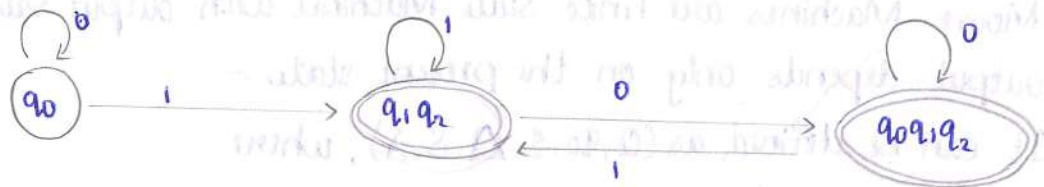
$$\delta(\{q_0, q_1, q_2\}, b) = \{q_1, q_2\}$$

The next will be the new state from the transition rules of previous state

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

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

Step-3: Construct the DFA from transition table



From the NFA, the final state is q_2 .

Hence in DFA, all states with q_2 are the final states

$$Q = \{q_0, q_1, q_2, q_0q_1, q_1q_2, q_0q_1q_2\}$$

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

$$q_0 = q_0$$

$$F = \{q_1, q_2, q_0q_1, q_1q_2, q_0q_1q_2\}$$

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

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

$$\delta(\{q_1, q_2\}, 0) = q_0q_1, q_1q_2$$

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

$$\delta(\{q_0q_1, q_1q_2\}, 0) = \{q_0q_1, q_1q_2\}$$

$$\delta(\{q_0q_1, q_1q_2\}, 1) = \{q_1, q_2\}$$

	0	1
q ₀	q ₀	q ₁ , q ₂
q ₁ , q ₂	q ₀ q ₁ , q ₁ q ₂	q ₁ , q ₂
q ₀ q ₁ , q ₁ q ₂	q ₀ q ₁ , q ₁ q ₂	q ₁ , q ₂
q ₀ q ₁ q ₂	q ₀ q ₁ q ₂	q ₁ , q ₂

Moore Machine.

Moore Machines are Finite State Machines with output values, and its output depends only on the present state.

It can be defined as $(Q, q_0, \Sigma, \Delta, \delta, \lambda)$, where

Q is a finite set of states

q_0 is the initial state

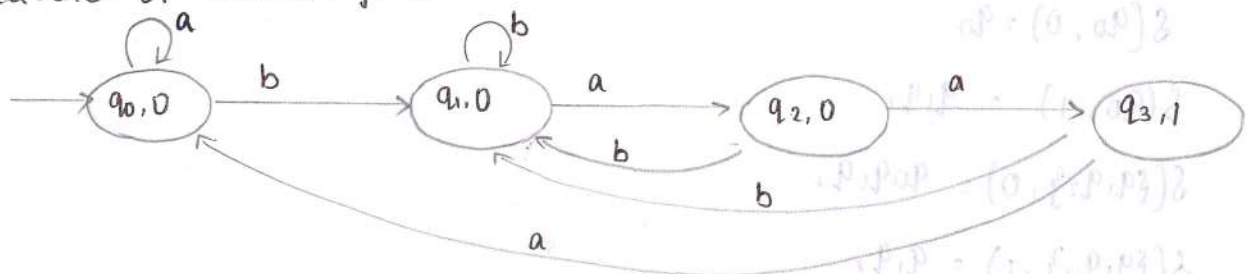
Σ is the input alphabets/symbols

Δ is the output alphabets/symbols

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

λ is the output function mapping from Q to Δ

Construct a Moore Machine that takes set of all strings over a, b and no. of occurrence of substrings baa



Transition Table.

	a	b	λ
q_0	q_0	q_1	0
q_1	q_2	q_1	0
q_2	q_3	q_1	0
q_3	q_0	q_1	1

For considering the occurrence of substring baa, we can take any strings like aaaaa, aabaa, abbaa, . . .

For example, aabbbbaabaa.

$$S(q_0, aabbbbaaaba) = 0$$

$$S(q_0, abbbbaaaba) = 0$$

$$S(q_0, bbbbaaaba) = 0$$

$$S(q_1, bbaaaba) = 0$$

$$S(q_1, baaba) = 0$$

$$S(q_1, aaba) = 0$$

$$S(q_2, aaba) = 0$$

$$S(q_3, aaba) = 0$$

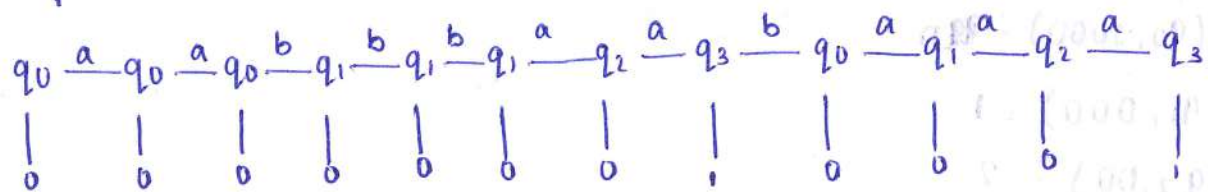
$$S(q_0, baa) = 0$$

$$S(q_1, aa) = 0$$

$$S(q_2, a) = 0$$

$$S(q_3) = 1$$

output :



$$\text{input} = n(aabbbbaaaba) = 11$$

$$\text{output} = n(000000010001) = 12$$

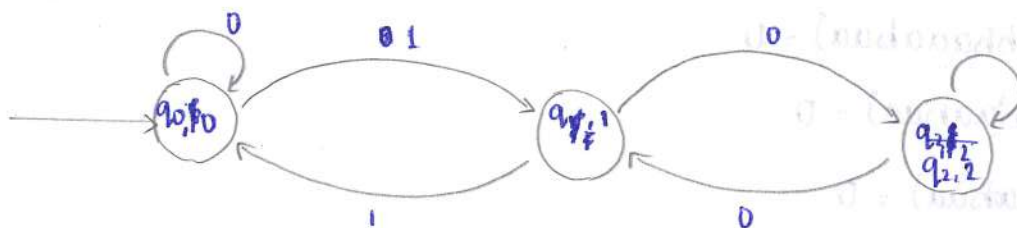
[In Moore machine, when the input length is 'n', the output length will be 'n+1']

Example :

Transition Table for a Moore Machine

states \ input	s = 0	s = 1	λ
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

The transition diagram for the above table of Moore Machine can be represent as:

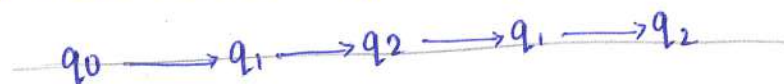


→ The initial state q_0 is marked with an arrow. The table define S & λ
 $S \rightarrow$ inputs, $\lambda \rightarrow$ output

→ In Moore machine, the output depends upon present state and not on present input

→ For the input string 1000

The transition states given by



$$S(q_0, 1000) = 010$$

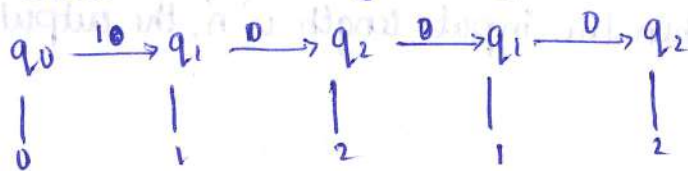
$$S(q_1, 000) = 1$$

$$S(q_2, 00) = 2$$

$$S(q_1, 0) = 01$$

$$S(q_2) = 2$$

The transition states given by



input = 1000

output = 01212

Mealy Machines

Mealy Machines are also finite state machines with output value, and their output depends on the present state and current input symbol. It can be defined as $(Q, q_0, \Sigma, \Delta, \delta, \lambda)$ where

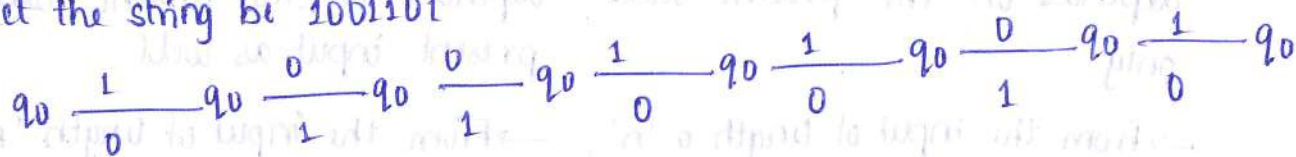
- Q is a finite set of states
- q_0 is the initial state
- Σ is the input alphabet/symbols
- Δ is the output alphabet/symbols
- δ is the transition function which maps $Q \times \Sigma \rightarrow Q$
- λ is the output function which map $Q \times \Sigma \rightarrow \Delta$

Construct a Mealy Machine that takes binary no. as input and gives 1's complement of that number as output. Assume the string is read from LSB to MSB.



input \ state	0		1	
	state	output	state	output
q_0	q_0	1	q_0	0

Let the string be 1001101

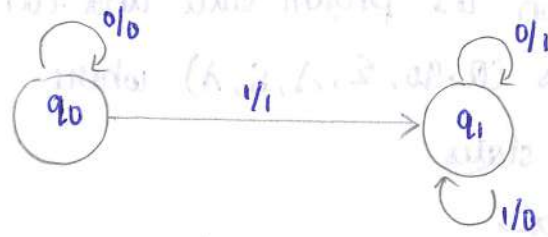


input = $n(1001101) = 7$

output = $n(0110010) = 7$

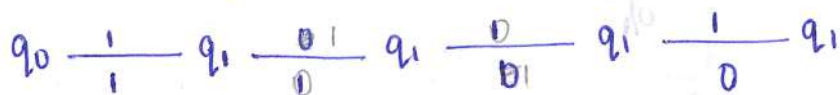
[In Mealy Machine, when the input length is 'n', the output length will also be 'n']

Construct a Melay Machine, that takes binary as input and produces 2's complement as output. Assume the string is read LSB to MSB



input \ state	0		1	
	state	output	state	output
q ₀	q ₀	0	q ₁	1
q ₁	q ₁	1	q ₀	0

Let the string be 1011



Differences between Moore & Melay.

Moore Machine	Melay Machine
→ The output of Moore Machine depends on the present state only	→ The output of Melay Machine depends on the present state and present input as well
→ From the input of length 'n', the output will be of length 'n+1'	→ From the input of length 'n' the output will be of length 'n'

Conversion of Moore to Melay Machine

In a Moore Machine, the output is represented on the state itself. In contrast, in a Melay Machine, the output is represented on the transition, or in other words, on the arrow itself.

Compared to the Melay-to-Melay conversion, this Moore-to-Melay conversion is relatively simple.

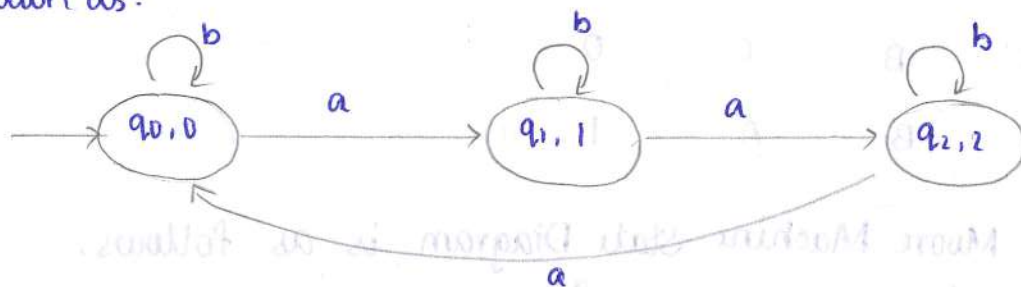
Converting a Moore Machine to a Melay Machine is simple. The key is to transfer the output information from the states in the Moore diagram to the arrows in the Melay diagram.

Example - 1

Transition Table for Moore.

input states	a	b	o/p
q_0	q_1	q_0	0
q_1	q_2	q_1	1
q_2	q_0	q_2	2

The Moore Machine state diagram for the above transition table can be drawn as:

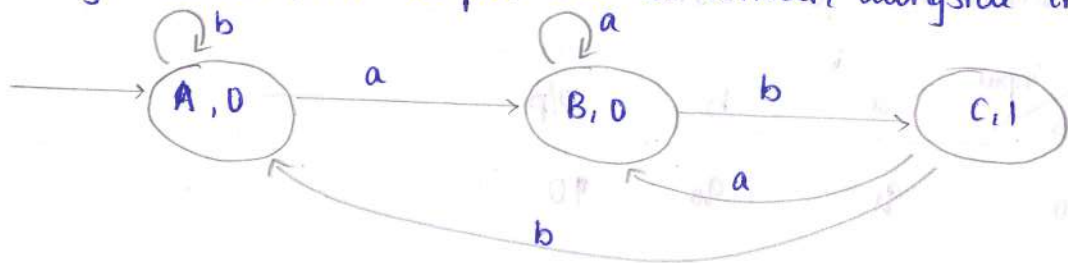


The transition table for Melay Machine can be constructed as follows:

	a	O/p	b	O/p
q_0	q_1	1	q_0	0
q_1	q_2	2	q_1	1
q_2	q_0	0	q_2	0

The ~~Moore~~ ^{Melay} Machine state diagram can be represented, using the transition table.

In Melay machine, the output will be written alongside the states

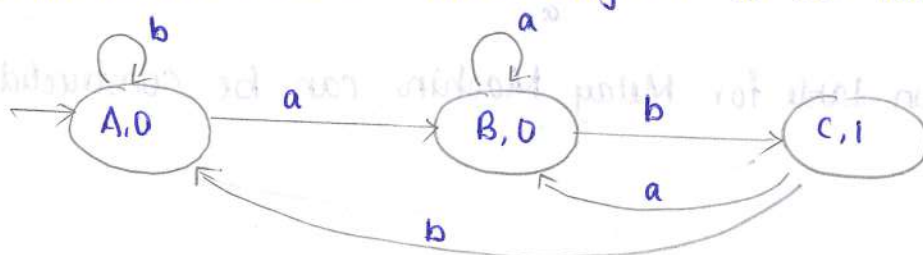


Example 2:

Transition Table for Moore

	a	b	O/p
A	B	A	0
B	B	C	0
C	B	A	1

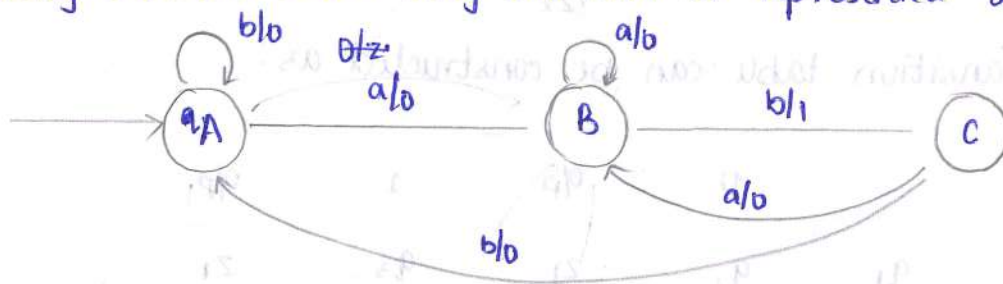
The ~~the~~ Moore Machine State Diagram is as follows.



Transition Table for Melay Machine

states input state	a	o/p	b	o/p
A	B	0	A	0
B	B	0	C	1
C	B	0	A	0

The Melay Machine status diagram can be represented as



Conversion to Melay to Moore

The conversion of a Melay machine to a Moore machine involves transforming a status machine where the output depends on both the current state and the input into a machine where the output depends only on the state.

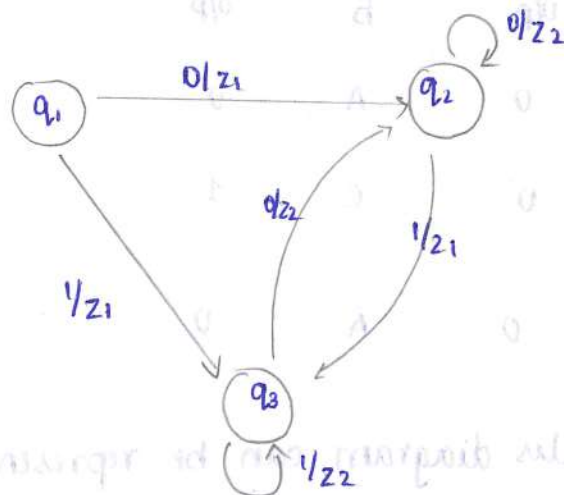
Since, the output of a Mealy machine changes immediately with an input change, while a Moore machine changes only on state transitions, we need to modify the states accordingly.

For perform conversion from Melay to Moore machine :

- Create a New state for each unique output
- Assign outputs to the new states
- Update State transition
- Define the Final moore machine

Example :

Let us consider a Melay Machine.

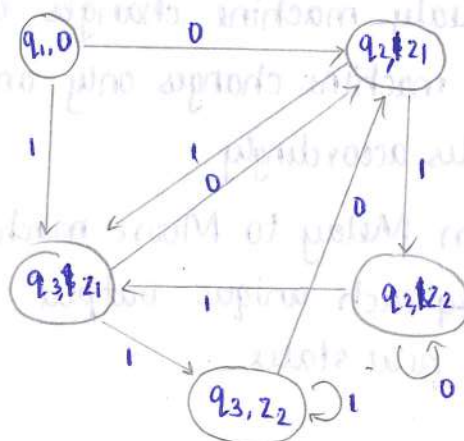


The transition table can be constructed as:

	0	q/p	1	q/p
q ₁	q ₂	z ₁	q ₃	z ₁
q ₂	q ₂	z ₂	q ₃	z ₁
q ₃	q ₂	z ₁	q ₃	z ₂

Constructing Moore Machine :

→ consider every unique output to have a new state.



	0	1	O/p
q_1	q_2	q_3	
q_2	q_2	q_3	z_1
q_2	q_2	q_3	z_2
q_3	q_2	q_3	z_1
q_3	q_2	q_3	z_2

