Finite Automata

- What is Toc ?

→ In Theoretical computer science the theory of Computation is the branch of computer science that deals with the mathematical problems and efficiently problems can be solved on model of compiting using an algorithm.

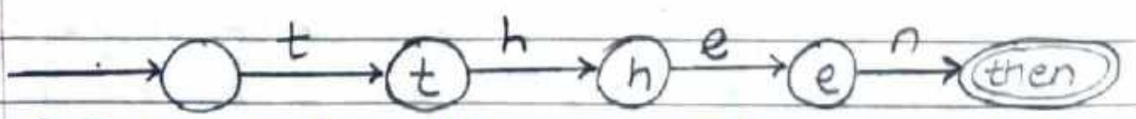
→ The theory of computation is divided in three parts:-

a] Automata
b] Computability
c] Computational Complexity

Automation Theory

★ What is the Automation Theory?

→ In Theoretical computer science automata is the study of the abstract machines and compititional problem that can be solved using the machines called as an automata theory.



A finite automation modelling recognition of then

Computation

→ Computation is a general term for any type of information processing that can be represented as an algorithm mathematically.

→ For example :-
We are adding two numbers in the calculator and converting a decimal number to binary representation and finding the addition of two numbers.

★ Basic Concepts

- Symbols

→ They are building block of language.

→ The members of the alphabet are the symbols of the alphabet.

→ We generally use capital greek summation (Σ) letter to designate the alphabet.

Example →

{a, b, c, d, e ..... 1, 2, 3, 4 ..... }

★ Alphabet

→ Alphabet is finite non empty set of symbol
→ We can use the greek summation (Σ) letter to show the alphabet
→ The alphabet over which the strings are defined may vary with the application.

★ Strings

→ Strings are non separated by comma
→ In that string we are using alphabet like (0 to 9) and (a to z)

For example
01101 where $\Sigma = \{0, 1\}$
abrdcadabr where $\Sigma = \{a, b, c \ldots z\}$

★ Empty String

→ The string with zero occurences of symbols from Σ and is denoted 'e'.
→ The empty string is denoted by 'e'
→ An empty string length is zero

Example
$\Sigma = \{\}$

★ Length of string

→ Length of string 'W' is usually written as |W|
Example → $|0101| = 4$
$|e| = 0$

★ Reverese

→ Reverse string is denoted by W^R

For example

if $W = abc$ then reverse string is $W^R = cba$

★ Concatenation

Concatenation means we are merging two strings

① For example

$X = a_1 a_2 ---- a_m$

$Y = b_1 b_2 ----- b_n$

So the

$xy = a_1 a_2 ---- a_m b_1 b_2 ---- b_n$

② For example

$X = 01101$

$Y = 110$

$XY = 01101110$

★ Substring

Substring means it is the subset of string and any string of concecutive characters in some string

For example

The string V is subset of W string if $W = abc$ then $V = a, ab, c$ are substring of W.

★ Language

Language means the set of string. The set of string are selected from number of alphabet language is subset of Σ^*

Example

1] $\Sigma = (a, b)$ L1 is length is two

$L1 = \{aa, bb, ab, ba\}$

2] $\Sigma = (a, b)$ L2 is length is three

$L2 = \{aaa, bbb, baa, abb, aba, bba, aab\}$

3] $\Sigma = (0, 1)$ the set of string consisting no. as 0's followed by no of 1's

$\{e, 01, 0011, 000111, \ldots\}$

4] $\Sigma = (0, 1)$ the set of string with equal no of 0's & 1's

$\{e, 01, 10, 0011, 0101, 1010, 1001, 1100\}$

★ Empty Language

Empty Language is denoted by $\emptyset = \{\}$

The empty language means their is no string are included and also not included the empty string has a several property.

Language can be finite and infinite :-

$L = \{a, aba, bba\}$

- the first eg is finite language because the string are provided directly in a language

$L = \{a^n \mid n \geq 0\}$

- the second eg the n^{th} value are provided means the n value are so it greatere than zero the condition is satisfied. If $\Sigma = (a,b)$ then L3 is a language of all strings are starts with a
$L3 = \{a, aa, aaa, aaaa, ...ab, abb, aab ....\}$

So, In this eg the string are starts with a so it is infinite no. of string are available.

★ Kleene Closure :-
- It is power of an alphabet (Σ^k)
- K is the powere of an alphabet.
- If L is a language then L^* is the set of all finite string formed by concatenating words from L so any word can be used any numbere of times including zero times.
- So, e also member of L^* means empty set is also number of a kleene closure.

eg :- $\Sigma = \{0,1\}$
$\Sigma^0 = \{e\} = 2^0$ Set of all string is length zero
$\Sigma^1 = \{0,1\} = 2^1$ set of all string is length one
$\Sigma^2 = \Sigma.\Sigma = \{0,1\}\{0,1\}$
$= \{00, 01, 10, 11\} = 2^2$ Set of all string is length 2
$\Sigma^3 = \Sigma^2 \cdot \Sigma = \{00, 01, 10, 11\} \cdot \{0, 1\}$
$= \{000, 010, 100, 110, 001, 011, 101, 111\} = 2^3$
Set of all the string is length is three

$\Sigma^* = \Sigma^\wedge = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3$

The set of all possible strings over Σ is called kleene closure denoted by Σ^*

★ Positive Closure :-

- If L is a Language then L+ is the set of all finite string formed by concatenating words from L. So any word can be used one or many times.
- The kleene closure and positive closure are same but minor difference is the kleene closure starts from zero and positive closure starts from 1

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

② $\Sigma^+ = \Sigma^+ \cup \{\varepsilon\} +$ [In these example it is included zero

③ $\Sigma^+ = \Sigma^* - \{\varepsilon\}$ [In these example zero is excluded]

	Operations	Definition
①	Union of L and M written as LUM	$LUM = \{s \mid s$ is in L or s is in M$\}$
②	Concatenating of L and M written as LM	$LM = \{st \mid s$ is in L and t is in M$\}$
③	Kleene closure of L written as L^*	$L^* = L^0 \cup L^1 \cup L^2 \cup \ldots\ldots$ L^* denotes "zero or more" concatenation of L

④ positive closure of L written as L^+	$L^+ = L^1 \cup L^2 \cup \ldots\ldots$ L^+ denotes "one or more concatination of $L^+ = LL^*$

Example 1

☆ Let $X = \{a, b, c\}$ and $Y = \{abb, ba\}$, Then

- $XY = \{aabb, babb, cabb, aba, bba, cba\}$
- $X^0 = \{e\}$
- $X^1 = X = \{a, b, c\}$
- $X^2 = XX = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$
- $X^3 = XXX = \{aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, bac, bba, bbb, bbc, bca, bcb, caa, cab, cac, cba, cbb, cbc, cca ccb, ccb, ccc\}$

Example 2

- The language $L = \{a, b\}$ consists of the strings over $\{a, b\}$ that contain the substring bb
- $bb \in L$
- $abb \in L$
- $bbb \in L$
- $abbb \in L$
- $bbaaa \in L$
- $bbabba \in L$
- $abab \notin L$ [It is not a language)

- bab $\notin$ L
- b $\notin$ L

Example 3

Let L be the language that consists of all strings that begin with aa or end with bb

- $L1 = \{aa\}, \{a, b\}^*$
- $L2 = \{a, b\}^* \{bb\}$
- $L = L_1 \cup L_2 = \{aa\}\{a, b\}^* \cup \{a, b\}^* \{bb\}$
- bb $\in$ L
- abb $\in$ L
- bbb $\in$ L
- aabb $\in$ L
- bb aaa $\notin$ L
- abab $\notin$ L
- bab $\notin$ L
- ba $\notin$ L

☆ Formal Language

Formal Language is based on Formal Rules

1. List of Symbols
2. Rules for forming a string
3. Rules for forming a sentence.

- The basic building block of a formal language is a alphabet rules for forming a string from alphabet are defined without only among any ambiguity.
- The list of legal string is called as formal language.

☆ Finite State Machine / System

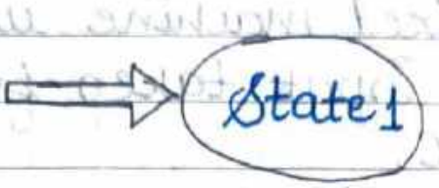
- FSM is a model of computation. It is a proce of sequence input that changes the state of the system.
- When all input is processed we observed the system final state to determine whether the inp sequence was accepted or not.

- Components of a finite state Machine :-

1. State of known state

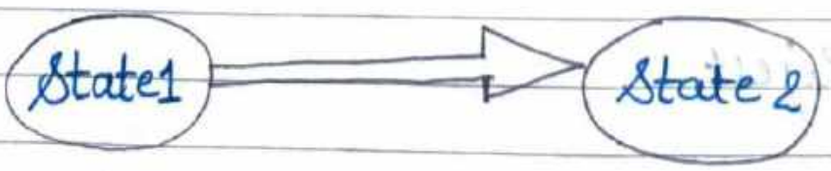
As the name implies there must be finite amount of states our system can be in a attive at a time

2] Initial state
The starting point of our systems so it is denoted as a arrow.



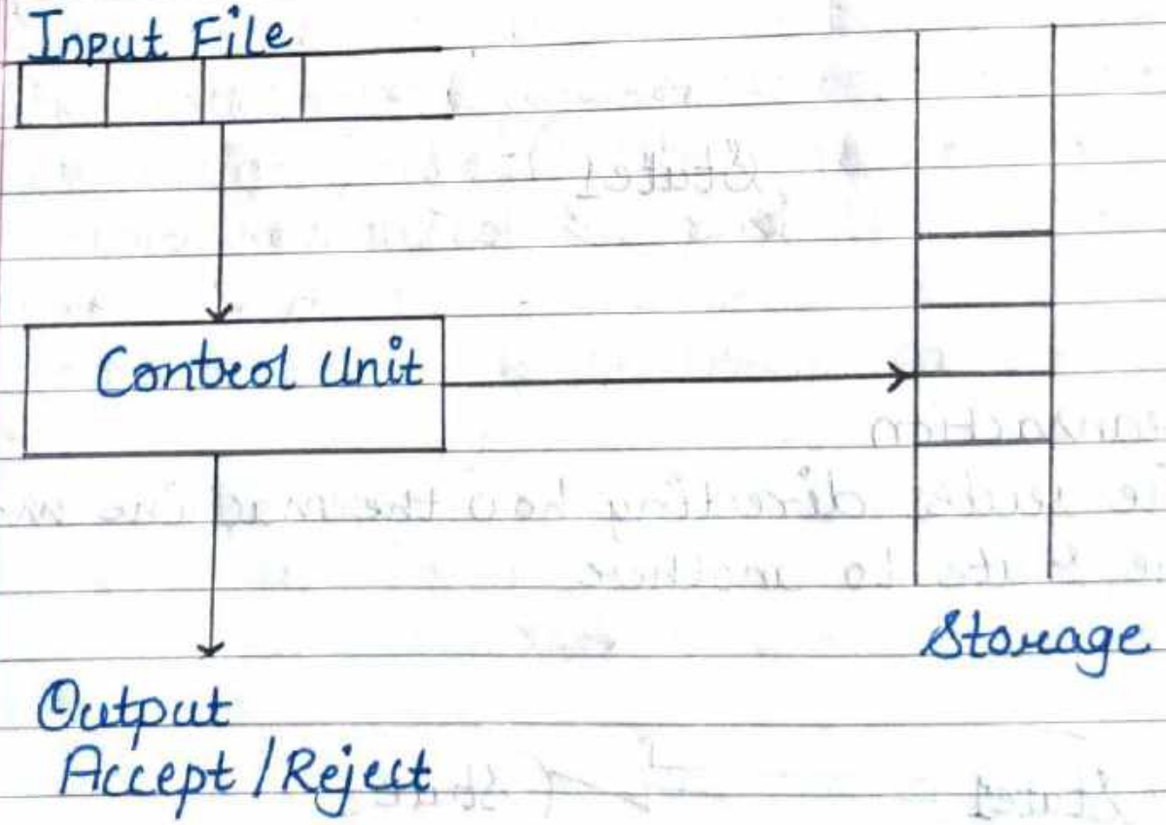
3] Set of accepting state
A subset of known states that indicates whether the input we processed is valid or not accepting states are usually drawn as a double circle.

4] Transaction
The reules directing how the machine move from one state to anothere



Finite Automata

- An automata is also known as FA [Finite Automata] or Finite State Machine [FSM]
- It is an abstract model of a digital computer.
- Is simple idealized machine used to recognize patterns within input taken from some characters set C.
- The job of a FA is to accept or reject an input depending on whether the pattern defined by the FA occurs in the input.



★ Input File

Input will be provided in tape format, the tape is divided in number of square or cell, each square is folded into single alphabet.

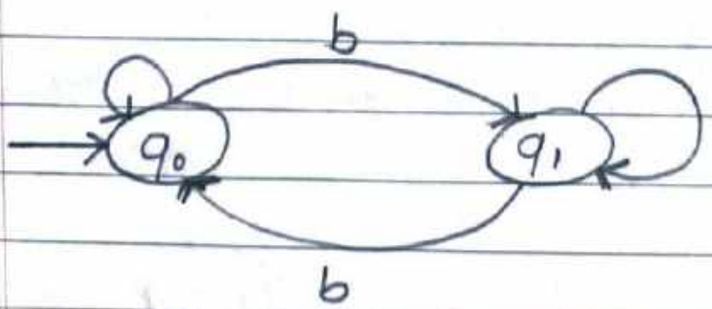
★ Control Unit

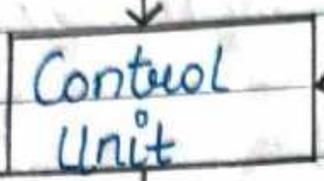
It has finite number of interenal state
It can in anyone of the interenal states can change the state in some define manner.

★ Storage

The storage aree stored temporary input
It consist an unlimited numbere of cells each cell can hold a symbol from an alphabet which can be a different from the input alphabet.

Example





Output
Rejected.

Storage

- **Limitation**
  - It cannot be used for computation
  - It cannot modify its input
  - It cannot used for context free Language
  - It has a finite number of states and it cannot recognize string of the from $a^n b^n$

Deterministic Finite Atomata

- In DFA for each input symbol one can determine the single state to which the machine will move, hence it is called as Deterministic Automata
- As it has a Finite number of state, the machine is called DFA
- The veretices represents states
- The arcs ladled with an input alphabet show the teansition
- A DFA can be represented by a 5-tuple $[Q, \Sigma, \delta, q_0, F]$ where

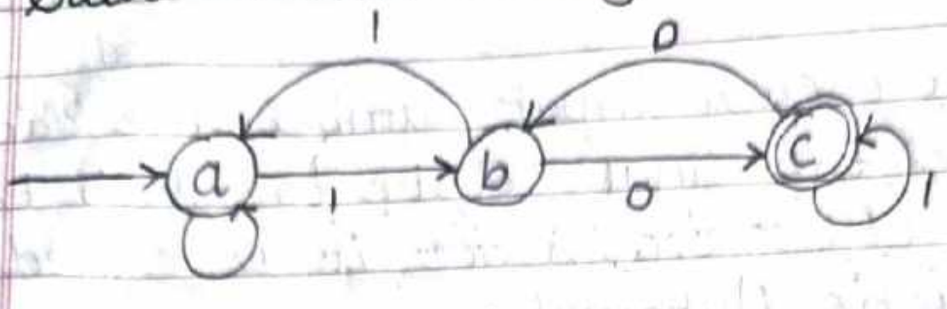
① Q is a finite set of states.
② Σ is a finite set of symbols called the alphabet
③ δ is the teansition function where $\delta : Q \times \Sigma \to Q$
④ q_0 is the initial state from where any input is processed ($q_0 \in Q$)
⑤ F is a set of final state / states of Q ($F \subseteq Q$)

Example

Let a deterministic finite automation be :-
Where $M = (Q, \Sigma, \delta, q_0, F)$ where
$Q = \{a, b, c\}$
$\Sigma = \{0, 1\}$
$q_0 = \{a\}$

$F = \{c\}$

★ State transition diagram.



★ State transition table.

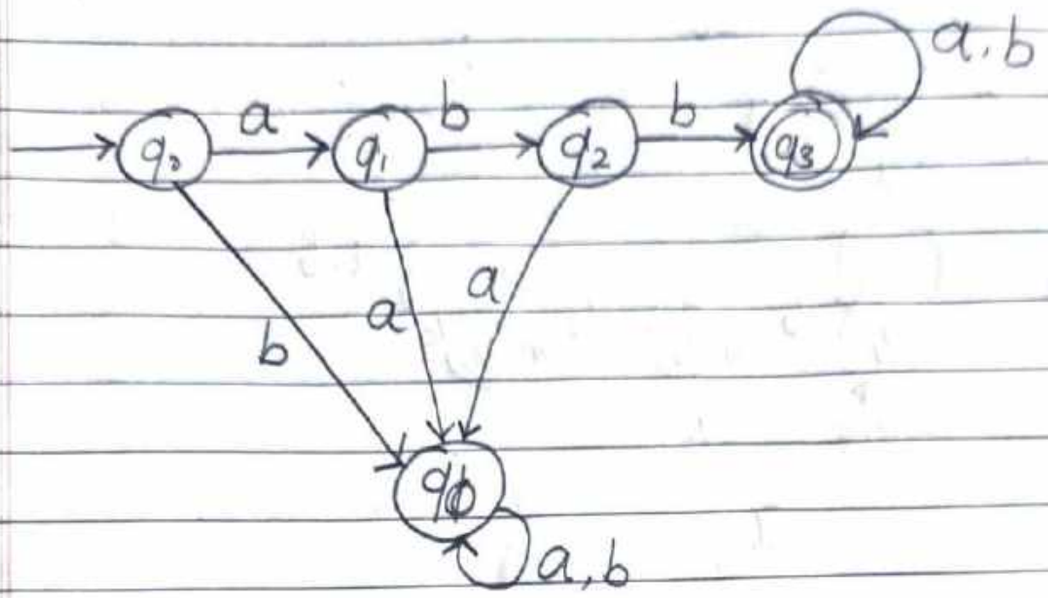
Present State	Next State for Input 0	Next State for Input 1
a	a	b
b	c	a
c	b	c

Example of DFA

Draw DFA for the following language over the alphabet {0, 1}

① All Strings staveting with abb,
② All strings with abb as substring
③ All strings ending in abb

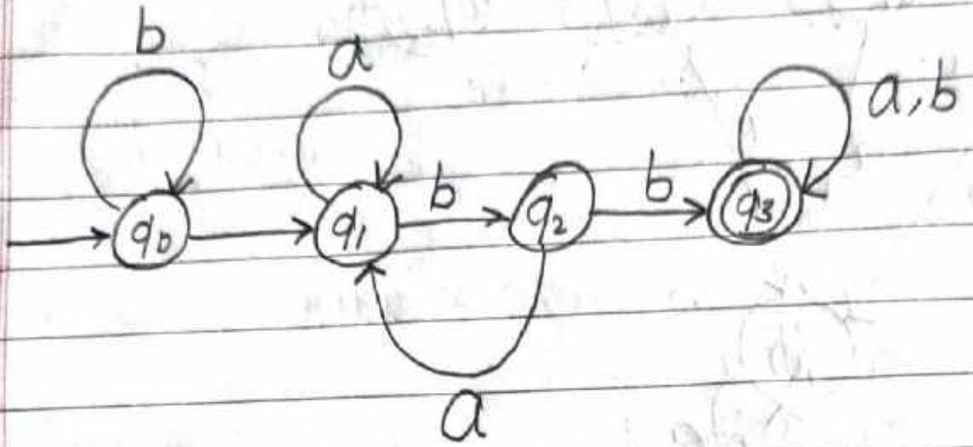
① All strings starting with abb



Transition Table

	a	b
q_0	q_1	ϕ
q_1	ϕ	q_2
q_2	ϕ	q_3
q_3	q_3	q_3

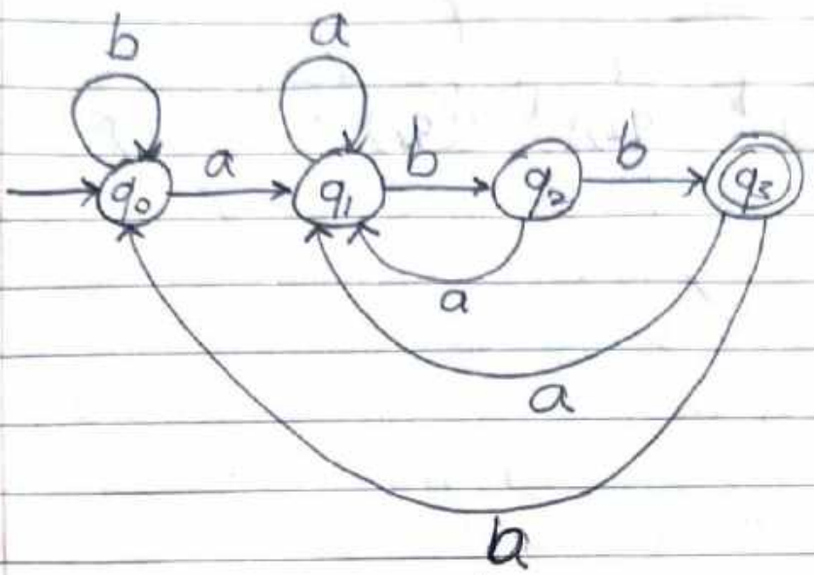
(2) Draw the transition diagram all string with abb as substring



Transition Table

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

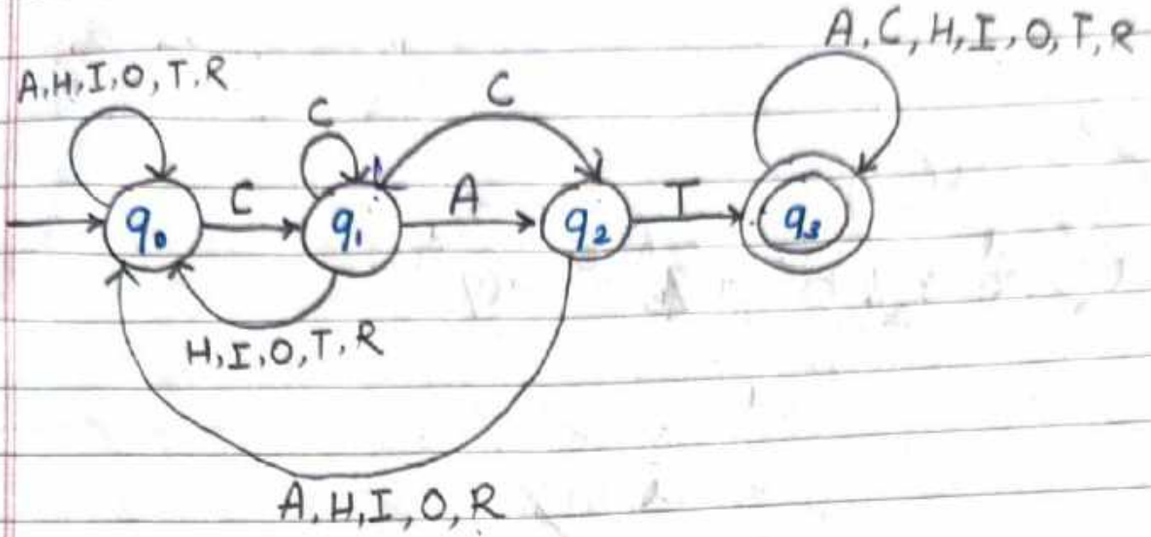
③ All strings ending in abb



Transition Table

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

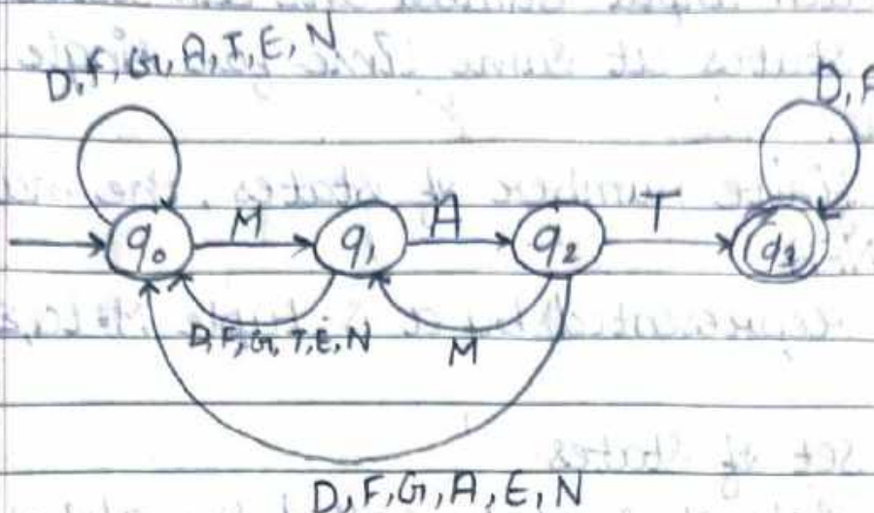
Q Design a DFA that read strings made up of letter in the word CHARIOT and recognize this strings that contains the word cat as a substring



State transition diagram.

Input	C	H	A	R	I	O	T
q_0	q_1	q_0	q_0	q_0	q_0	q_0	q_0
q_1	q_1	q_0	q_2	q_0	q_0	q_0	q_0
q_2	q_1	q_0	q_0	q_0	q_0	q_0	q_3
q_3	q_3	q_3	q_3	q_3	q_3	q_3	q_3

Q DFGMATEN, Substring MAT



★ Non-Deterministic Finite Automata (NFA)

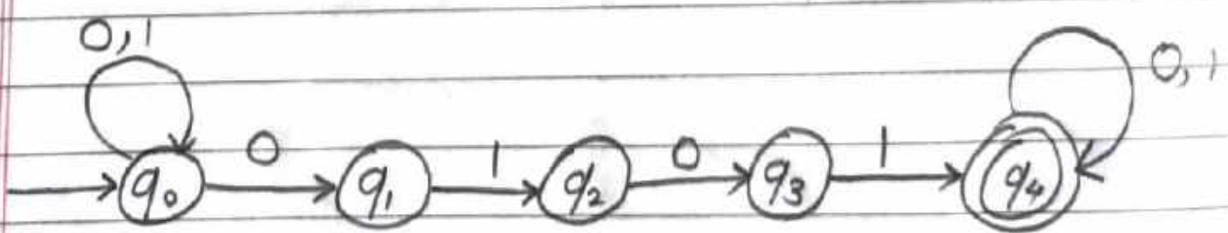
- In NFA for each input symbol one can determine the multiple states at same time for single input symbol.
- As it has a finite number of states, the machine is called as NFA
- A NFA can be represented by a 5-tuple $M = \{Q, \Sigma, \delta, q_0, F\}$ where:-

a] Q is a Finite set of States
b] Σ is a finite set of symbols called the alphabet.
c] δ is the transition function where $\delta = Q \times \Sigma \rightarrow Q$
d] q_0 is the initial state from where any input is processed ($q_0 \in Q$)
e] F is set of final state / States of Q ($F \subseteq Q$)

Examples:-

Q1] Draw NFA to accept strings containing the substring 0101

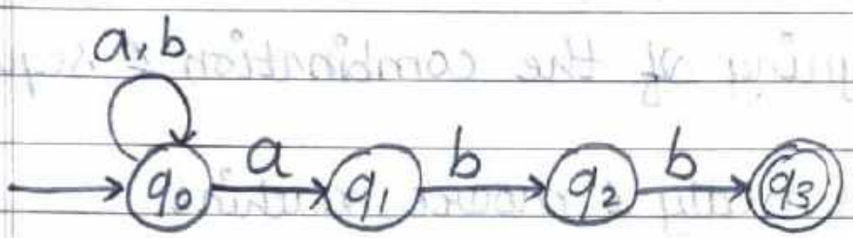
State transition diagram.



Transition Table

i/p Symbol	0	1
q_0	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
q_2	q_3	ϕ
q_3	ϕ	q_4
q_4	q_4	q_4

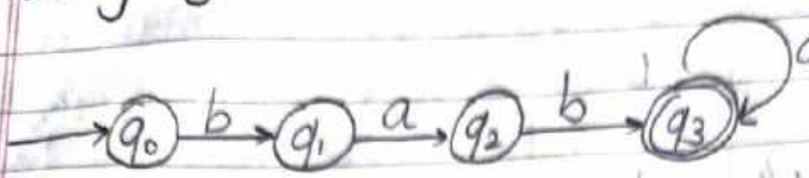
Q2] Draw NFA that accept various string from language $L = \{x \in \} \{a, b\}$ ends with abb



Transition Table

Input	a	b
q_0	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
q_2	ϕ	q_3
q_3	ϕ	ϕ

Q Language $L = \{x \in \}\{a, b\}$ start with bab.



- Application of Finite Automata

1] For the designing of lexical analysis of a compiler

2] For the recognizing the patteren using regular expression.

3] For the designing of the combination & sequentia

4] Circuits using mealy & moore machine.

5] Used in text editors

6] For the implementation of spell checkers.

∘ Finite Automata as Output Device (Moore and Mealy Machine)

- Finite Automata can also be used as an output device.
- Such machine don't have final state.
- Such machine can generates o/p on every input
- The value of o/p is a function of current state current input.
- This machine can be characterized by two behaviours:-

a] State transition Function (STF): denotes by δ

$\delta : \Sigma \times Q \rightarrow Q$

b] Output Function / machine Function (MAF): denoted by λ

$\lambda : \Sigma \times Q \rightarrow O$

$\lambda : Q \rightarrow O$

∘ State transition diagram For Mealy Machine

- A mealy machine is a Finite state machine whose o/p values are determined by current state & current input.

- A mealy machine can be represented by 6 tuple $M = (Q, \Sigma, O, \delta, \lambda, q_0)$ where

1] Q is a Finite set of states

2] Σ is a Finite set of input symbols called the alphabet

3] O is a Finite set of output symbols.

4] δ is the transition function where

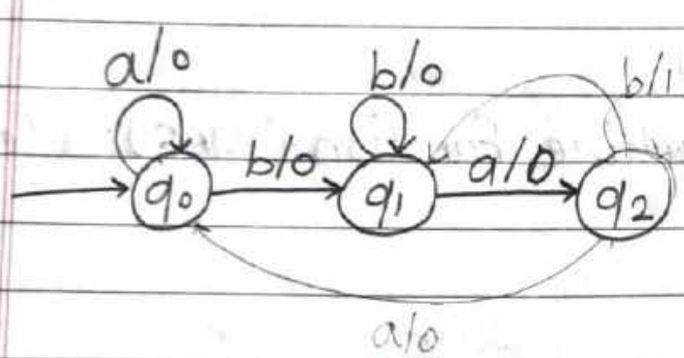
$\delta : Q \times \Sigma \rightarrow Q$

5] λ is the output function where $\lambda: Q \times \Sigma \rightarrow Q$

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

Example

1] Construct mealy machine that gives output of 1 if the input string end with 'bab'



Transition Table

Input Symbols	a	b
q_0	$q_0/0$	$q_1/0$
q_1	$q_2/0$	$q_1/0$
q_2	$q_0/0$	$q_1/1$

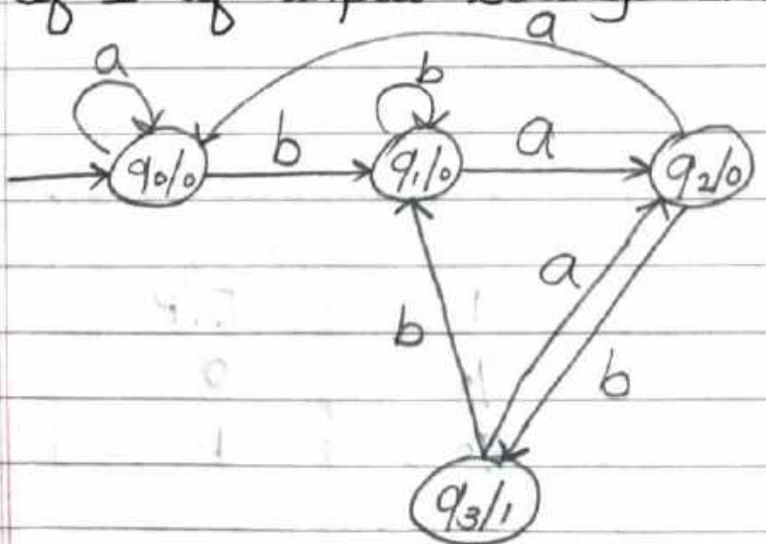
- Formal Definition of Moore Machine :-
- A Moore machine is a Finite State Machine whose o/p values are determined by the current state only.
- A Moore machine can be represented by a 6-tuple.

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

- Q is a Finite set of states.
- Σ is a Finite set of input symbols called the alphabet.
- O is a Finite set of output symbol.
- δ is a transition function where $\delta : Q \times \Sigma \rightarrow Q$
- λ is the O/p function where $\lambda : Q \rightarrow O$
- q_0 is the initial state from where any input is processed ($q_0 \in Q$)

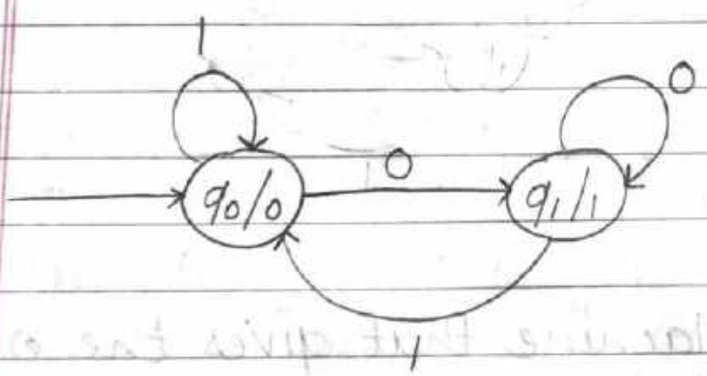
Example :

1] Construct Moore Machine that gives the output of 1 if input strings end with 'bab'



Input	a	b	I/p
q_0	q_0	q_1	0
q_1	q_2	q_1	0
q_2	q_0	q_3	0
q_3	q_2	q_1	1

Q Construct Moore Machine that gives output of 1's compliment of binary number

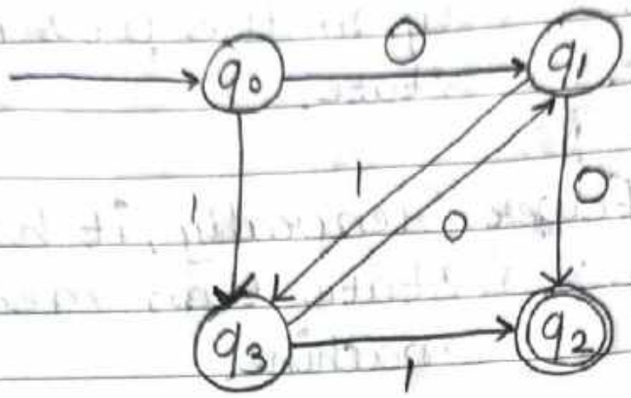


	0	1	I/P
q_0	q_1	q_0	0
q_1	q_1	q_0	1

★ Difference between Mealy and Moore Machine

Mealy	Moore
1] Output depends upon the present state & the present input.	Output depends only upon the present state.
2] Generally, it has fewer states than Moore machine.	Generally, it has more state than mealy machine.
3] The value of the output function is a function of the transitions & the changes when the input logic on the present state is done.	The value of the output function is a function of the current states & the changes at the clock edges whenever state changes occur.
4] Mealy machine react faster to inputs. They generally react in the same clock cycle.	In moore machine more logic is required to decode the output resulting in more circuit delays. They generally react one clock cycle later.

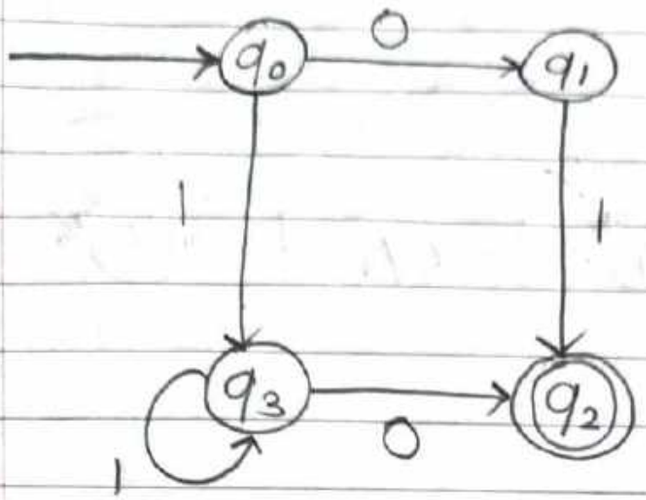
Q3] Design a DFA to accept string ending with 00 or 11.



Transition Table

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

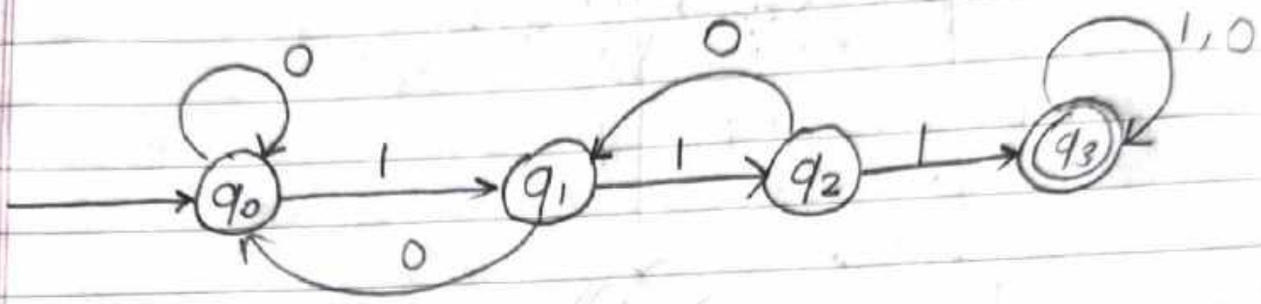
Q4] Design a DFA to accept string ending with 01 or 10



Transition Table.

	0	1
q_0	q_1	q_3
q_1	q	q_3
q_2		
q_3	q_2	q_3

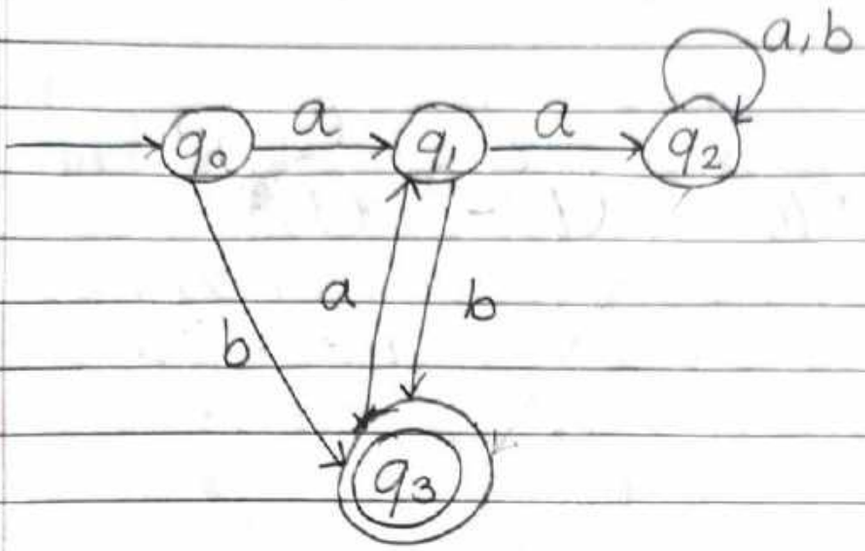
Q5] Design a DFA for accepting L over Σ 0, 1 such that every substring of length 4 contains at least three one's.



Transition Table

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

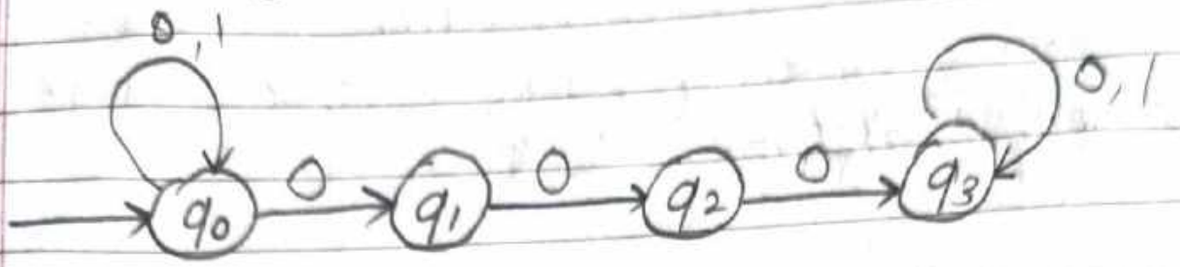
Q6] Design a DFA that accept the string ending with b & not containing aa over $\Sigma = \{a, b\}$



Transition Table

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

Q7) Design NFA for language over $\Sigma = \{0, 1\}$ containing all string have conjucative three 0

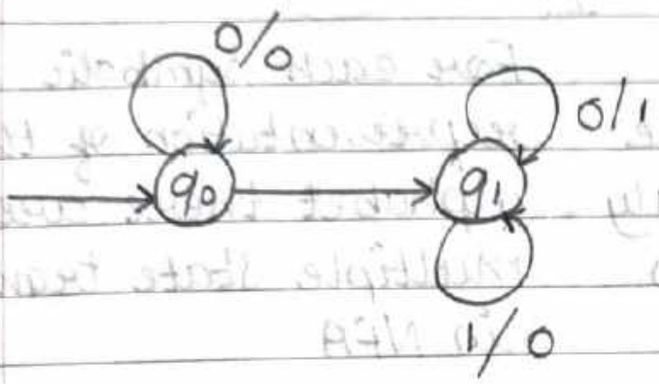


Transition Table

	0	1
q_0	q_1, q_0	q_0
q_1	q_2	ϕ
q_2	q_3	ϕ
q_3	q_3	q_3

Q8] Construct a mealy machine for 2's compliment

Binary number is 1011
Then, it's 1's compliment is 0100
and 2's compliment is 0101



	0	1
q_0	q_0	q_1
q_1	q_1	q_1

Difference between DFA and NFA

DFA	NFA
1] DFA is Deteeministic Finite Automata	NFA is Non-Deteermini Finite Automata.
2] For each symbolic reepresentation of the alphabet theree is only one state teansition in DFA	For each symbolic reepresentation of the alphabet theree aee multiple state teansitio in NFA
3] DFA can be understand as one machine	NFA can be understand as multiple little machines computing at same time.
4] DFA can not use empty string teansition	NFA can use empty string teansition
5] In DFA, the next possible State is distinctly set	In NFA each pair of State & input have many possible next state.
6] DFA is moee difficult to constreuct	NFA is easieo to constreuct
7] Time needed for executing an input string is less	Time needed for execu an input string is more.

8] All DFA are NFA	Not all NFA are DFA
9] DFA requires more space	NFA requires less space than DFA.
10] It is easy to determine whether we as transitions are deterministic	It is difficult to determi whether $W \in L$ as transitions are non-deterministic.