# MODULE-1

## Introduction to Automata Theory :-

Automata Theory is a study of Finite state Machines (FSM) or Finite Automata. Finite Automata are the abstract machines that perform essential functions of Software or hardware. That is before we develop certain hardware or Software, we develop Finite Automaton model and test this model by providing all possible inputs. If the model work fine then we proceed to develop actual Software or Hardware.

states: 1. off
2. on

examples (i):

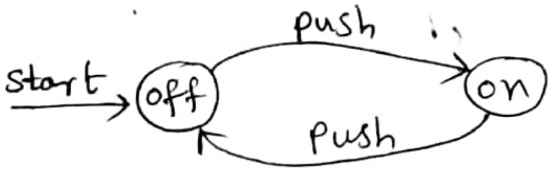



Fig:- Finite State Machine Model for ON/OFF Switch

example (ii):

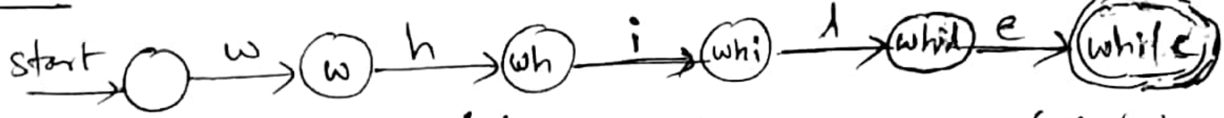



Fig: Finite State Machine model to recognize 'while' Keyword of 'C' Language

Finite state Machine (Finite Automata) Consist of Finite number of states. Each state is represented by a Circle. Transistion from one state to another is represented by drawing a directed arc labelled with the input symbol, the one of the state is designated as a "start state", the state in which the system is placed initially. In the example 1, Start state is "off". There may be one or more "final" or "accepting" states. Enter into these states indicate that the input suplied to Finite automata is valid.

## Why we study Automata Theory ?
(Applications of Theory of Computation)

Automata Theory Concept can be used to develop a model for some of the important Software or Hardware which are listed below.

1. Software for designing and checking the behaviour of digital circuits.
2. The "Lexical analyzer" of a compiler, that is the compiler component that breaks the input text into logical units such as identifiers, keywords operators, punctuation symbols
3. Software for scanning collections of Web pages, to find occurences of words, phrases, or other patterns.
4. Software for verifying systems of all types that have a finite number of states, such as the communication protocols or protocols for the secure exchange of information.

## Central Concepts of Automata Theory :-

1. Alphabet
2. String
   - Length of a string
   - empty string
   - power of an alphabet
   - Concatination of two strings
3. Language
4. Problem

Note: Symbol :- is any single character whether it is letter or digit or any of other character example. 'a', '4', '+', A ';' etc

(1) Alphabet :- An alphabet is a finite set of Symbols. We use the symbol $\Sigma$ (Sigma) for an alphabet.

Examples: (1) $\Sigma = \{0, 1\}$ – set of binary digits

(2) $\Sigma = \{a, b, .., z, A, B, .. Z, 0, 1, .. 9, +, -, *, (, ), \{, \}, ;, ', ', ....\}$

– C language alphabet

(3) $\Sigma = \{a, b, .., z\}$ – set of lower case letters

(2) String (Word) :- A string is a finite sequence of symbols all of which are chosen from some alphabet. For example 01101 is a string over alphabet $\Sigma = \{0, 1\}$. 'while' is a string over the Alphabet of 'C' Language.

013412 is not a string over alphabet $\Sigma = \{0, 1\}$ since symbols 3, 4 & 2 are not chosen from $\Sigma$.

Empty string :- is the string with no symbols. Empty string is denoted by epsilon ($\epsilon$).

Length of a string :- is the number of positions of symbols in the string. Length of string w is denoted by |w|. For example Length of string

w = 01101 is $|w| = |01101| = 5$

positions of symbols 1 2 3 4 5

Note: string is denoted by w, & length of string w is denoted by |w|.

Concatenation of two strings x & y is a new string xy formed by appending string y to the string x.

for example string $x$ = Computer & $y$ = Science

then $xy$ = Computer Science.

Power of an alphabet :- is the set of all strings of certain length. We use $\Sigma^K$ to define the set of all strings of length K.

example :- If $\Sigma = \{0,1\}$ is an alphabet, then

$\Sigma^0 = \{\epsilon\}$ $\quad \Sigma^1 = \{0,1\}$ $\quad \Sigma^2 = \{00, 01, 10, 11\}$

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

$\Sigma^*$ :- Let $\Sigma$ be an alphabet. $\Sigma^*$ can be defined as the set of strings of all the lengths.

for example, Let $\Sigma = \{0, 1\}$ be an alphabet

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

$$\Sigma^* = \{\epsilon\} \cup \{0,1\} \cup \{00, 01, 10, 11\} \cup \{000, 001, 010, 011, 100, 101, 110, 111\} \cup \ldots\ldots\}$$

$$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \ldots\ldots\}$$

(3) Language :- A Language is a set of strings all of which are chosen from $\Sigma^*$, where $\Sigma$ is a particular alphabet. If $L \subseteq \Sigma^*$ then L is a language over $\Sigma$.

example (i) $L = \{0, 00, 000, 10, 110, 0110, 1110, \ldots\ldots\}$

set of all strings that end with 0.

(ii) $L = \{10, 11, 101, 111, \ldots\ldots\}$ (2, 3, 5, 7)

Set of all strings which are prime numbers in binary.

(iii) $L = \{01, 0011, 000111, \ldots\ldots\}$

Set of all strings consisting of n number of of 0's followed by n no. of 1's.

(4) problem :- is a question of deciding whether a given string is a member of some particular Language. If $\Sigma$ is an alphabet and L is a language over $\Sigma$ then the problem is:-

Given a string w in $\Sigma^*$. decide whether or not w is in L.

example: Let $\Sigma = \{0, 1\}$ is an alphabet and L is a language consisting of all strings of 0's & 1's which are prime numbers

Given $L = \{10, 11, 101, 111, \ldots\ldots\}$ (2, 3, 5, 7)

& string $w = 1101$, if w is a prime number, then it is a member of the Language. otherwise Not.

A Formal Language can be described in one of the following ways:-

(1) As a Sentence in English

example: Set of all strings of 0's and 1's that ends with 0.

(II) In the form of Set Former notation:-

The Language is described as follows:-

$L = \{w \mid \text{Some description about string } w\}$

read as "the Language L Consist of Set of all strings w such that whatever appear to the right of Vertical bar.

example ① $L = \{ w \mid w$ Consist of $n$ number of 0's followed by $n$ number of 1's $\}$

(ii) $L = \{ w \mid w$ Consist of 0's and 1's that end with 0 $\}$

(iii) **As an expression notation** :- Here the string $w$ is replaced by an expression with parameters and to the right of Vertical bar, we specify Conditions on the parameters.

example ① : $L = \{ 0^n 1^n \mid n \geq 1 \}$

—"— (ii) $L = \{ 0^i 1^j \mid i \leq 0 \leq j \}$

## Finite state Machine (FSM) :-

Finite state Machine Can be Written in the form of a diagram as follows :-

input string

$W = 0101101$

$q_0, q_1, \ldots q_n$ - states

start state is $q_0$

Control Unit

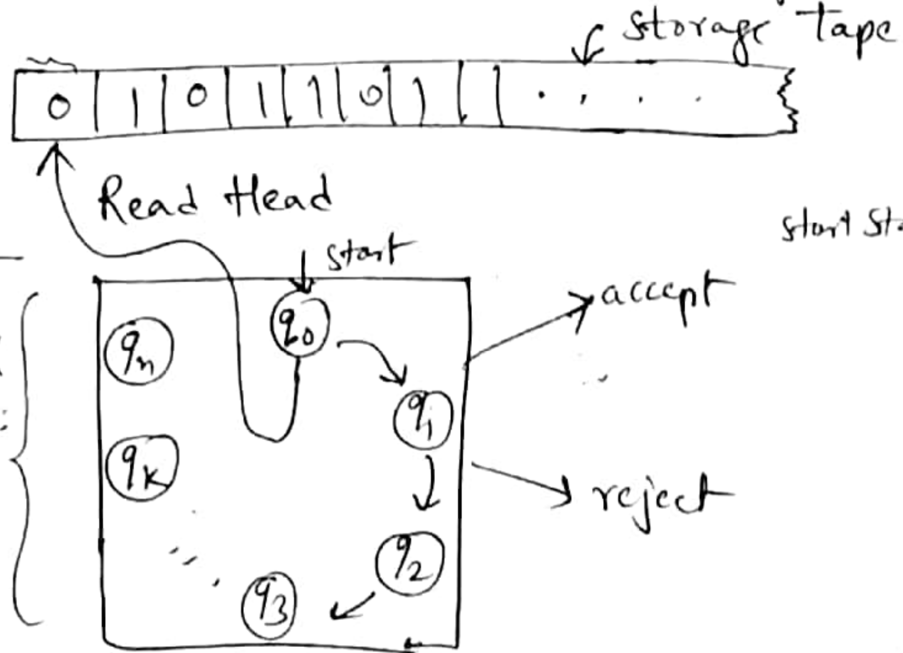



Figure (1) :

A Finite State Machine Consist of

① **Storage Tape** :- is a long tape which is divided into Cells, each Cell stores one input symbol, We store the input string $w$ on the tape.

The Finite state Machine Consist of

② **Control Unit** :- The Finite state Machine Consist of Finite number of states. Each state remembers the previous symbol that was read.

From the present state say $q_i$ it read one symbol $a_i$ of the tape and moves to next state $p_j$ i.e $\delta(q_i, a_i) = p_j$.

(iii) Output: It can be either accept or reject.

Finite state Machines can be classified into following types:-

(1) Deterministic Finite State Machine (DFSM)
(2) Non-Deterministic Finite State Machine (NDFSM or NFSM)
(3) $\epsilon$ - NDFSM (epsilon NDFSM)
(4) Push Down Automata (PDA)
(5) Turing Machine. (TM)

Working of Finite State Machine:- Initially, the Finite state Machine will be in the start state say $q_0$. Input string $W = 010101$ is stored on the storage tape and Read head points to first symbol say 0 of the input string (Figure (1)). Suppose $\delta(q_0, 0) = q_1$. The FSM goes to next state $q_1$ and Read head now moves so that it points to next symbol say 1 of the input string. Suppose $\delta(q_1, 1) = q_2$. The Finite Automata goes to next state $q_2$. In this way each time Finite automata read current symbol from each state and go on moving from one state to another state. Assume it has finished reading the symbols input string $W = 010101$ and halted in state $q_n$. If $q_n$ is a Final (accepting) state, then input string $W = 010101$ is accepted. If $q_n$ is a Non-Final state, the input string is rejected.

Language Hierarchy :- Formal Languages Can be classified into

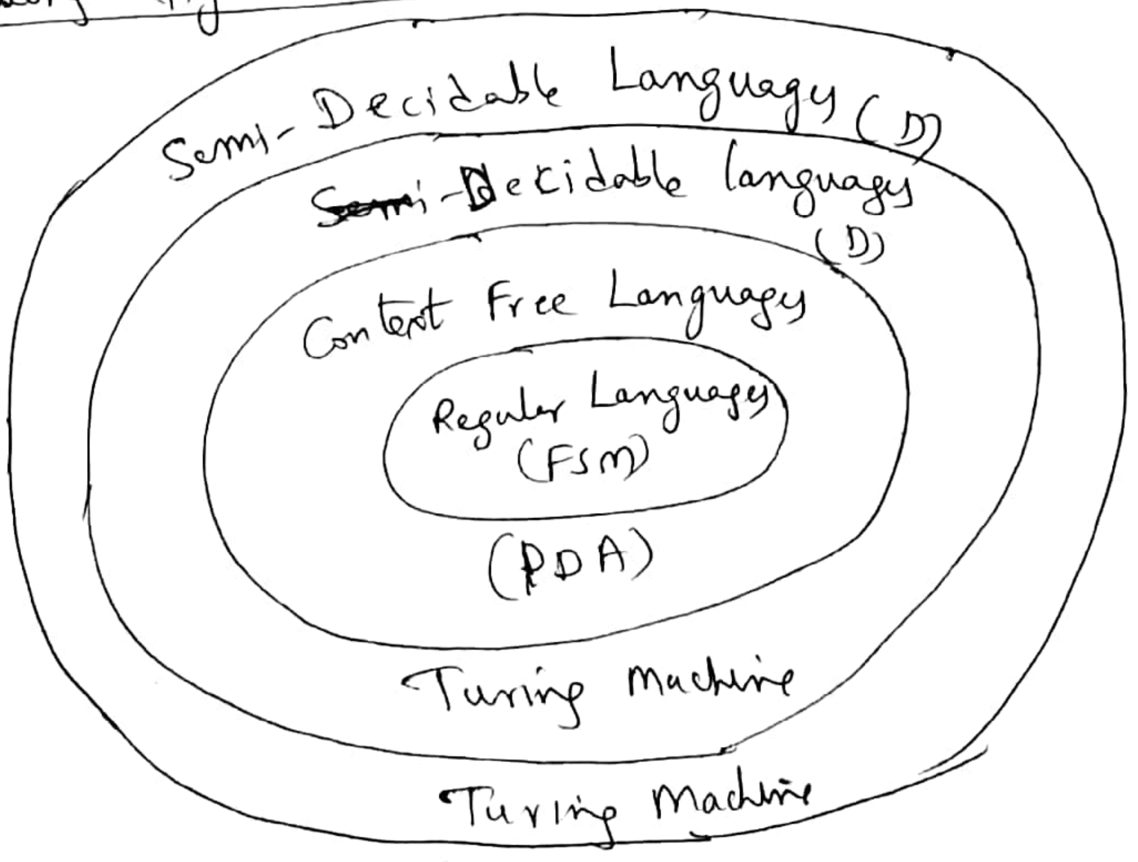
(1) Regular Languages, ~~which~~ Can be accepted by Some Finite State Machine (FSM)

(2) Context Free Languages, Which Can be accepted by PDA (Push Down Automata)
They can also be generated by Context Free Grammars.

(3) Decidable Languages (D), Which Can be decided by Some Turing Machine that always halts after accepting strings of language.

(4) Semi-decidable languages (SD), which can be semi-decided by Some Turing Machine that halts on all strings of language.

Following figure shows Hierarchy of formal languages



The best example of Finite Automaton is ON/OFF Switch shown in following figure.

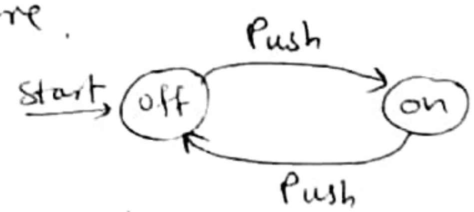



fig: Finite Automaton model for a ON/OFF Switch.

The Finite Automaton has two states: i) off and ii) on

The number of states ~~are~~ in a FA are Finite

~~First~~ initially the machine ~~is in~~ start state say off, on external influence (input) i.e if we press, then FA moves to on state.

## Deterministic Finite State Machines (DFSM) :- (Important)

* Abreviated as DFSM
* The term 'deterministic' means the DFSM ~~deter~~ can determine exactly the next state giv when it knows present state and current input symbol
* On reading each input symbol, DFSM moves to exactly one state

## Definition of Deterministic Finite State Machine (DFSM) :-

A DFSM consist of :-

1. A Finite Set of states denoted by ~~To~~ Q.
2. A Finite Set of input Symbols denoted by Σ.
3. A transistion function denoted by δ that takes as arguments a state and input symbol and returns a state.

   δ can be written as $\delta(q, a) = p$
   (Delta)

   q → present state, a → present input symbol, p → Next state

   δ can be represented graphically as :- $q \xrightarrow{a} p$ (Present state) (Next state)
4. A start state denoted by $q_0$
   DFSM is in start state before reading any input string.
5. A set of final or accepting states denoted by F.
   F is a subset of Q

$$\boxed{\text{A DFSM can also be represented as a tuple with 5 components} \quad A = (Q, \Sigma, \delta, q_0, F)}$$

## How a DFSM Processes strings :-

input: string → DFSM (DFA) → output: accept / reject.
Ex: 01010

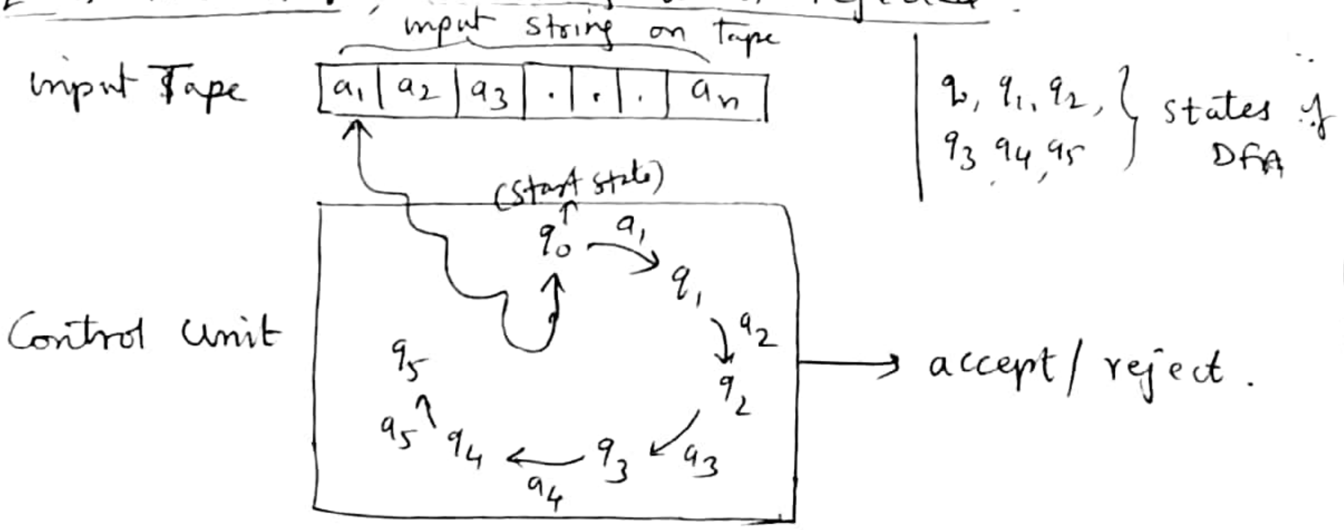
* It is required to understand how a DFSM decides whether to accept or reject a given input string.
* The Language of DFA is the set of all strings that the DFA accepts.

Suppose $a_1 a_2 \ldots a_n$ is a string which is given as input to the DFSM. The DFSM begins with start state $q_0$. Suppose it has transistion function $\delta(q_0, a_1) = q_1$. With this, DFA in state $q_0$, reads input symbol $a_1$ and moves to state $q_1$. $q_1$ is the state reached after reading input symbol $a_1$.

Let DFSM moves from $q_1$ to $q_2$ on reading input symbol $a_2$ using the transistion function $\delta(q_1, a_2) = q_2$. This process of reading next input symbol and moving to the next state continues until DFSM reads all the symbols $a_1 a_2 \ldots a_n$ of input string and enters some state $q$.

If $q$ is in F (set of final states), input string $w = a_1 a_2 \ldots a_n$ is 'accepted' by DFSM.

If $q$ is not in F, the string $w$ is rejected.




**LANGUAGE** :- We know that Language is a set of all strings, each of which are taken from $\Sigma^*$ where $\Sigma^*$ is a set of all strings of any length.

**Example 1**: Let $\Sigma = \{0, 1\}$ is an alphabet

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

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

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

The Language is denoted by L.

Let L is a set of all strings that ends with 1.

$$\therefore L = \{1, 01, 11, \ldots\}$$

L consist of set of all strings, each taken from $\Sigma^*$.

Therefore language is a Set of all strings, each string is taken from $\Sigma^*$. Normally $L$ is a Subset of $\Sigma^*$. i.e, $L \subseteq \Sigma^*$.

A language can be described in 3 ways.

(i) In the form of a sentence in English.

Ex:- Set of all strings of 0's and 1's with equal number of each. that is $L = \{01, 10, 1010, 0101, 1001, 0110, \ldots\}$

(ii) In Set Former Notation:- (Set Notation) In this notation, the language is described as follows:-

$L = \{w \in \Sigma^* : \text{Some description about string } w\}$

read as "Set of all strings $w$ such that whatever appear to the right of vertical bar.

examples: (i) $L = \{w \in \{0,1\}^* : w \text{ Consist of equal no. of 0's and 1's}\}$

(ii) $L = \{w \in \{0,1\}^* : w \text{ has } n \text{ no. of 0's followed by } n \text{ number of 1's}\}$

(iii) $L = \{w \in \{0,1\}^* : w \text{ is a binary integer that is prime number}\}$

Ex: $L = \{\overset{2}{10}, \overset{3}{11}, \overset{5}{101}, \overset{7}{111}, \ldots\}$

(iii) Expression Notation:- Here we replace $w$ by expression with parameters and describe strings in the language by stating the conditions on the parameters.

examples: (i) $L = \{0^n 1^n \mid n \geq 1\}$

read of set of all strings with $n$ no. of 0's followed by $n$ number of 1's such that $n \geq 1$.

Ex: $L = \{01, 0011, 000111, \ldots\}$

(ii) $L = \{0^i 1^j \mid 0 \leq i \leq j\}$

The Language has set of all strings with some zero's (may be no zero) followed by atleast as many 1's

Ex $L = \{\epsilon, 01, 0111, 011, 0011, 000111, \ldots\}$

NOTE:- Given a language $L$ that is described in one of the above three notations, we need to design DFA D that accepts those strings that are in the language and reject other ~~all~~ strings.

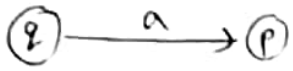
Notations for DFSM :- DFSM Can be represented in one of the following notations

(1) Transistion Diagram & (11) Transistion Table

(1) Transistion Diagram :- A Transistion Diagram for a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is a graph defined as follows.

a) For each state in Q there is a node (Circle)

b) For each state q in Q and each input symbol a in $\Sigma$, let $\delta(q, a) = p$. The transistion diagram has an arc from state q to state p labeled with symbol a.

That is 

c) There is an arrow into the start state $q_0$.

that is Start → $(q_0)$

d) Nodes Corresponding to final or accepting states are marked by a double Circle, states not in Set F have a single Circle,

Problem: (1) Design DFSM that accepts the following Language 'Set of all strings that Contain Substring 01'

Let L = {01, 01101, 00101, 1101, 11001, 101010101, ....}


fig: Transistion Diagram for DFA accepting all strings with a Substring 01.

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

where $Q = \{q_0, q_1, q_2\}$ $\Sigma = \{0, 1\}$ $\delta$ - As in Transistion Diagram

$q_0 = q_0$ (start state) $F = \{q_2\}$ (Set of final states)

(2) Transistion Table :- It is a Tabular representation of the Transistion functions $\delta$ (Delta) of Finite State Machine. all the Transistion functions of Finite State Machine. there rows of the table Corresponds to states, and the Columns Correspond to input symbols. The entry for the row for state q and Column for symbol a is the next state $\delta(q, a) = p$.

**Example:** following is the Transition Table representation of DFSM that accepts 'Set of all strings with a substring 01'

| $\delta$ | 0 | 1 |
|---|---|---|
| $\rightarrow q_0$ | $q_1$ | $q_0$ |
| $q_1$ | $q_1$ | $q_2$ |
| $* q_2$ | $q_2$ | $q_2$ |

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

present state, Current Symbol, Next state

## Extending the Transition Functions to strings :-

We know that Transistion function is denoted by $\delta$, that is $\delta(q, a) = p$. $\delta$ describes the next state reached by DFA from present state $q$ and on reading input symbol $a$.

(Delta) Present state, Present input Symbol, Next State

**Extended Transistion Function:-** is denoted by $\hat{\delta}$ that is (Delta Cap)

$\hat{\delta}(q, \underset{w}{01101}) = p$

$\hat{\delta}$ describes the state reached by DFA from start state $q$ and on Reading the input string $w$.

We define $\hat{\delta}$ by induction on length of string as follows:

**BASIS:-** $\hat{\delta}(q, \epsilon) = q$. DFA is in state $q$ and read no symbols and remain in state $q$

**INDUCTION:** Suppose string $w$ is of the form $xa$ where $a$ is the last symbol of $w$, and $x$ is a substring. For example $w = 1101$ is divided into $x = 110$ and $a = 1$. Now we compute

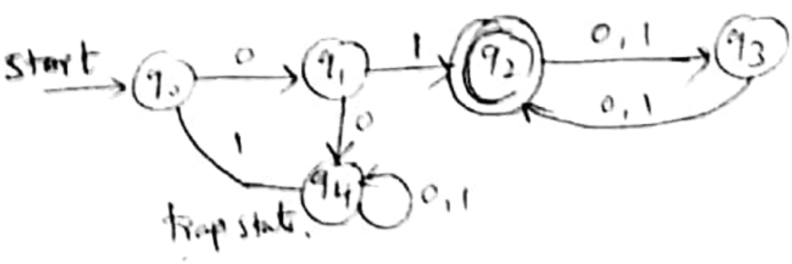
$$\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$$

That is to compute $\hat{\delta}(q, w)$, first we compute $\hat{\delta}(q, x)$ let it be $p$. Now we compute $\delta(p, a)$ which is the $\hat{\delta}(q, w)$.

**Problem(2):** Design DFA to accept the Language

$L = \{w \mid w \text{ is of even length and begins with } 01\}$

Transition Diagram:



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

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

$\Sigma = \{0, 1\}$ $\quad \delta$ = As in Transistion Diagram

$q_0 = q_0$ (start state)

$F = \{q_2\}$ set of final states. (accepting)

Consider the string $W = 010101$. Sequence moves made by DFA to accept $w$ is :-

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

Since $q_2$ is in $F$, 01010 is accepted

Consider another string. $W = 1010$

$q_0 \xrightarrow{1} q_4 \xrightarrow{0} q_4 \xrightarrow{1} q_4 \xrightarrow{0} q_4$. Since $q_4$ is not in $F$, $w = 1010$ is rejected.

**Important characterstics of Deterministic FSM :-**

(i) DFSM can determine exactly the next state when it is given.. with present state $q$ and present input symbol $a$.

(ii) Next state is always a single state

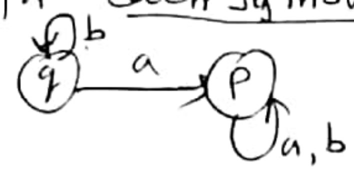
that is $\delta(q, a) = p$ ← Next state

(iii) DFSM has exactly one transistion (arc) out of any state for the same input symbol.

That is $q \xrightarrow{a} p$

(iv) DFSM is in only one state at any time $t$

(v) DFSM has one transistion for each symbol of alphabet out of any state, that is



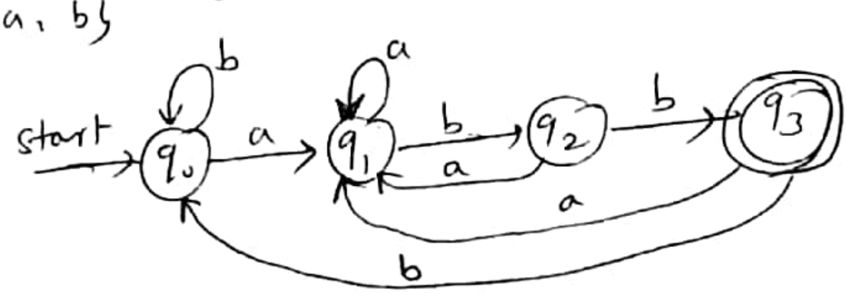

$\delta(q, a) = p$

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

## PROBLEMS

**Design DFSM for the following Languages :-**

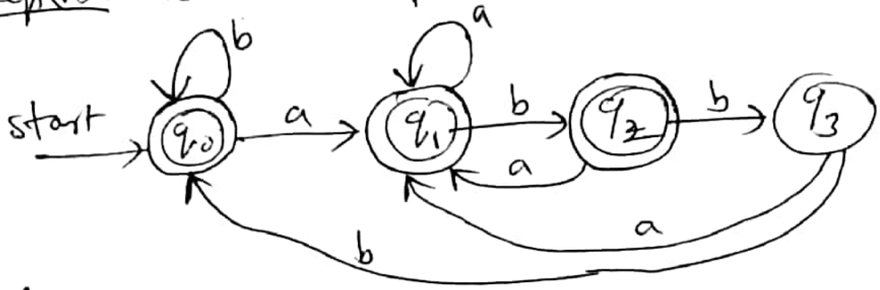
(1) Set of all strings that end with abb. over alphabet $\Sigma = \{a, b\}$



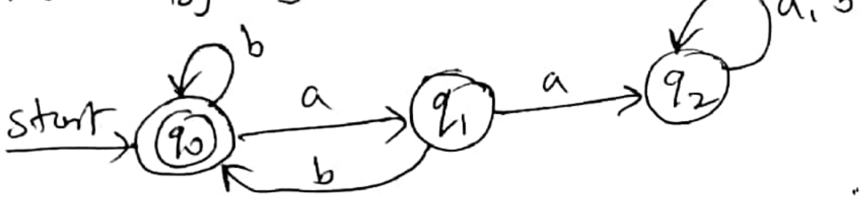
(2) $L = \{ w \in \{a,b\}^* :$ w do not end with abb $\}$

Step (1): Design DFSM Similar to problem (1)

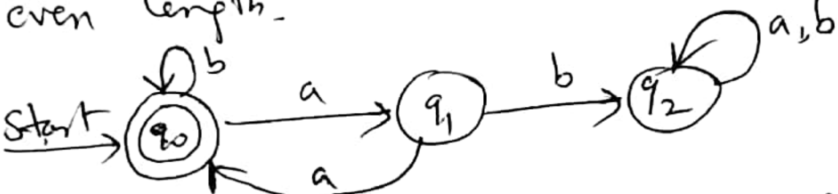
Step (ii): Write Compliment DFSM.




(3) $L = \{ w \in \{a,b\}^* :$ every a is immediately followed by b




(4) $L = \{ w \in \{a,b\}^* :$ every 'a' region in w is of even length.

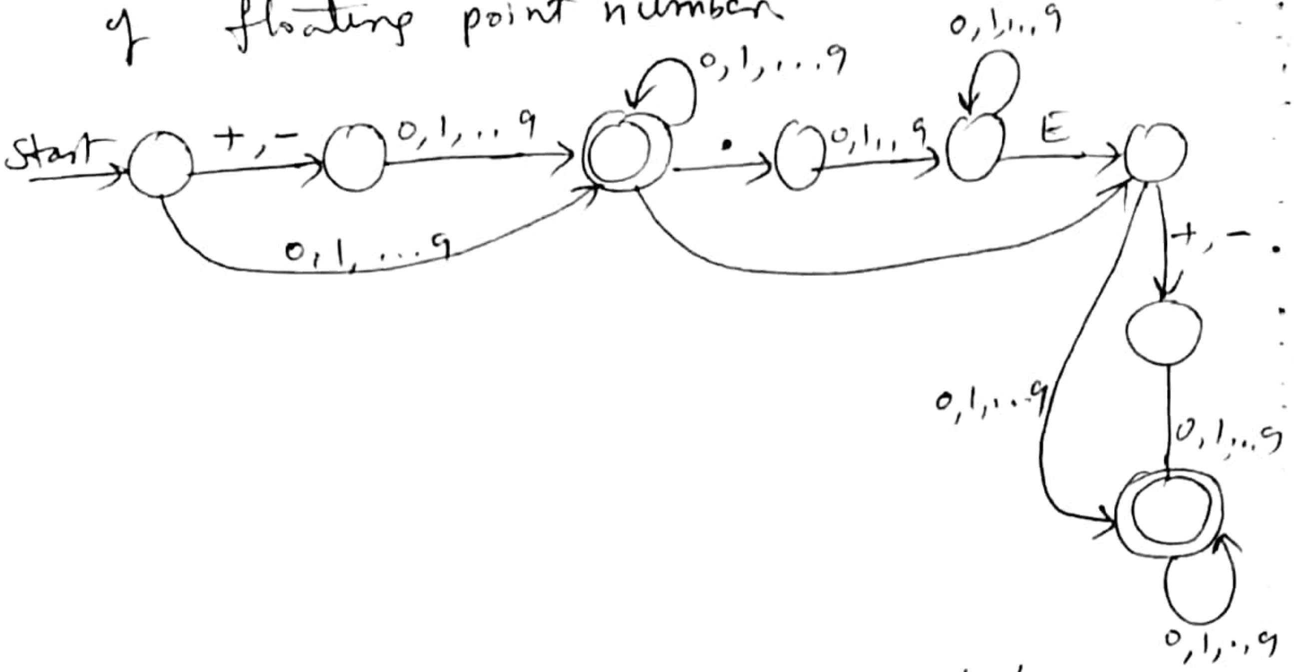



(5) $L = \{ w \in \{0,1\}^* :$ w has odd parity$\}$

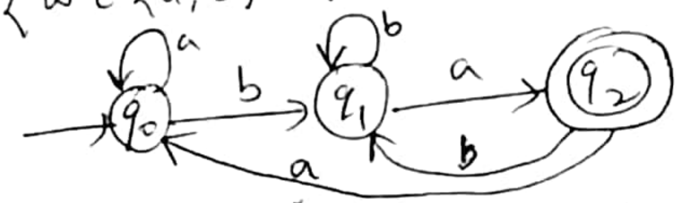
Note: A Binary String has odd parity iff number of 1's are odd Ex: 01101


(6) $L = \{ w \in \{a,b\}^* :$ w Contains no more than one b$\}$

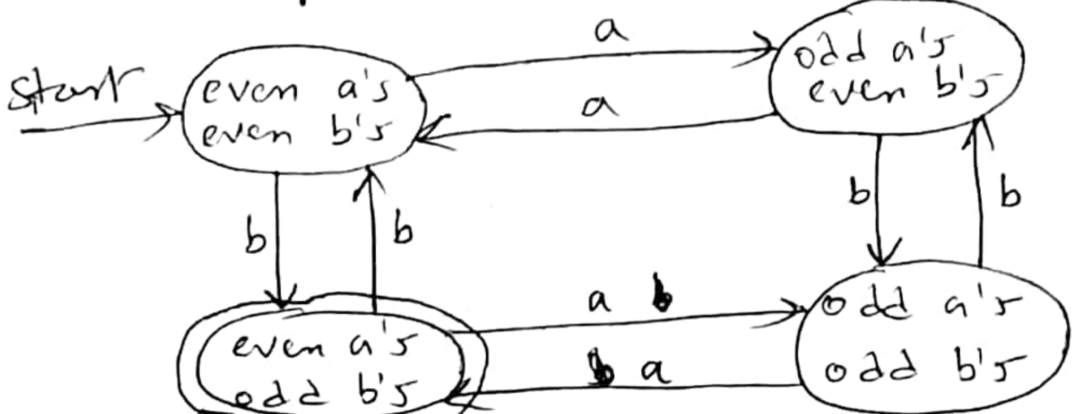
(7) $L = \{ w$ : $w$ is string representation of floating point number



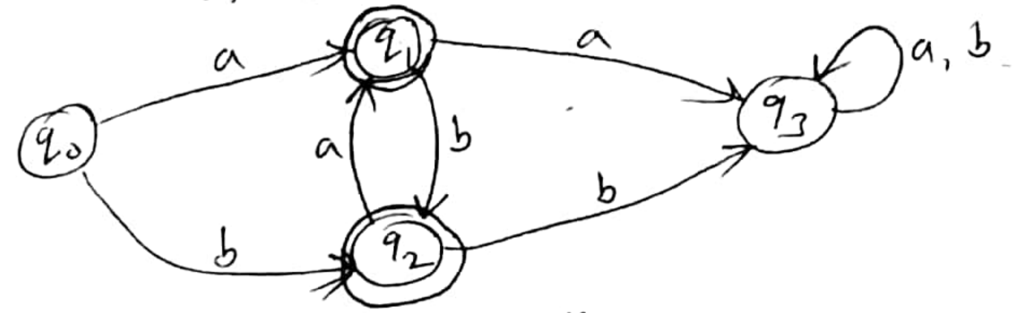

(8) $L = \{ w \in \{a, b\}^*$ : $w$ ends with $ba\}$




(9) $L = \{ w \in \{a, b\}^*$ : $w$ contains an even number of a's and odd number of b's




(10) $L = \{ w \in \{a, b\}^*$ : no two consecutive characters in $w$ must be same



## Non-Deterministic FSM's (NDFSM)

**Definition:** A NDFSM $N = (Q, \Sigma, \delta, q_0, F)$ is a quintuple consisting of

1. Finite set of states denoted by $Q$.
2. Finite set of input symbols denoted by $\Sigma$.
3. Transistion Function denoted by $\delta$ which takes present state and input symbol and returns a set of one or more states.

That is $\delta : Q \times \Sigma \rightarrow 2^Q$

In particular, $\delta(q, a) = \{p, q\}$ or $\{q\}$

Present state — input symbol — Next state

4. Start state denoted by $q_0$
5. Set of Final or accepting states, denoted by F.

**Example:** Consider the following NDFSM which accepts set of all strings that end with 01

Let $L = \{01, 110101, 000101, 1111001, \ldots\ldots\}$

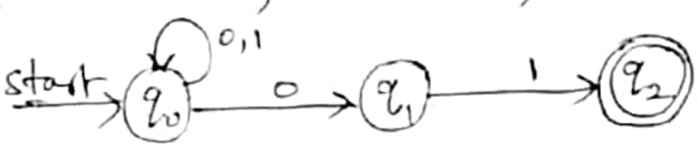
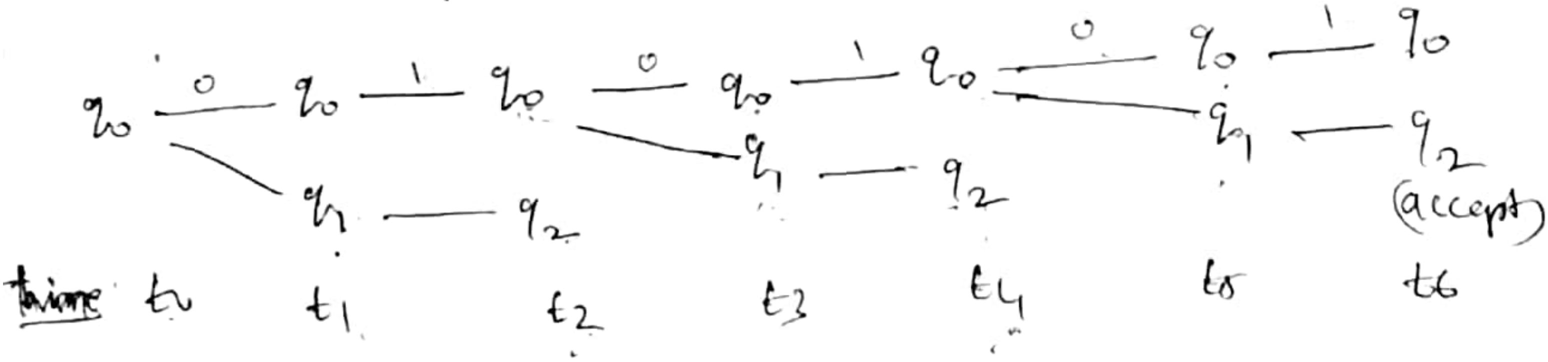



Fig:- Transistion Diagram of **NDFSM** accepting the above language

Sequence of state changes made by NDFSM while processing input string $w = 010101$ is



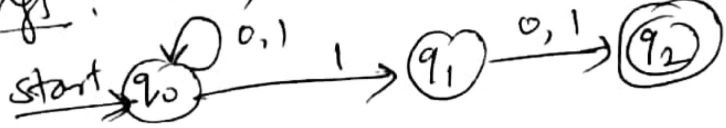
## Characterstics of Non-Deterministic FSM

* It may or may not determine the single next state given the present state & input symbol
* next state is a set of one or more states
* It has the ability to stay in several states at once
* it can ~~make~~ guess on the input symbols.
* On reading input symbol it may move to one or more next states.

**Design NDFSM for the following languages**

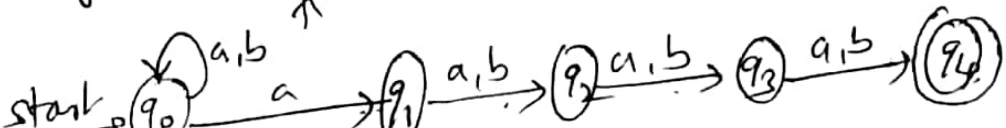
(1) $L = \{ w \in \{0,1\}^* :$ w contains 1 as second symbol from last.

Sample strings : 010, 011, 101010, 111010,

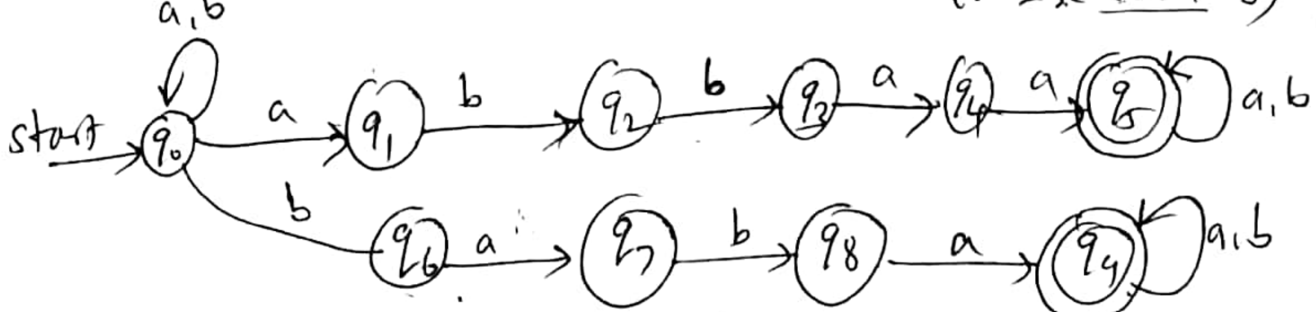



(2) $L = \{ w \in \{a,b\}^* :$ w contains ~~character~~ 'a' as fourth symbol from last $\}$

Sample strings : b a b a b a b ,  a b a a b a b ,




(3) $L = \{ w \in \{a,b\}^* : \exists\, x, y \in \{a,b\}^* ((w = x\ \underline{abbaa}\ y)$
$\vee\ (w = x\ \underline{baba}\ y)) \}$

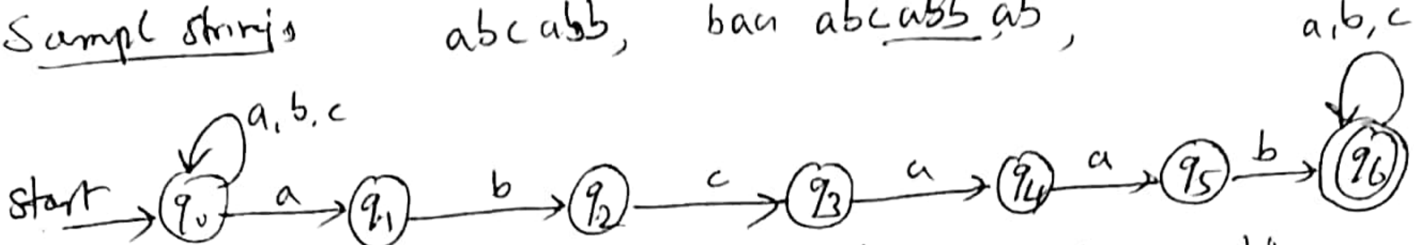



Sample strings : bab abbaabaa , ab babab a , . . . .

(4) $L = \{ w \in \{a, b, c\}^* : \exists\, x, y \in \{a, b, c\}^* \ (w = x\ abcaab\ y)\}$

- i.e. language $L$ consist of set of all strings $w$ containing atleast one occurence of substring abcabb.

Sample strings abcabb, baa abcabb ab,




## Non-Deterministic FSM which allows $\epsilon$ transistions (epsilon)

Also Known as $\epsilon$-NDFSM (epsilon NDFSM)

* It has the ability to move to a next state even without reading any input symbol.

That is $\delta(q, \epsilon) = \{p, q\}$

$q$ → present state; $\epsilon$ → Without reading any Symbol; $\{p, q\}$ → set of next states.


Definition of $\epsilon$-NDFA :— $\epsilon$-NDFSM $N$ is a five tuple $N = (Q, \Sigma, \delta, q_0, F)$ consisting of

(1) Finite set of states denoted by $Q$

(2) Finite set of input symbols denoted by $\Sigma$.

(3) Transistion Function denoted by $\delta$ (delta) which takes current state and either input symbol or $\epsilon$ (epsilon) and returns ~~set of~~ one or more next states. That is $\delta : Q \times \{\Sigma \cup \epsilon\} \rightarrow 2^Q$.

For example: $\delta(q, a) = \{p, q\}$

or $\delta(q, \epsilon) = \{p, q\}$

(4) start state denoted by $q_0$

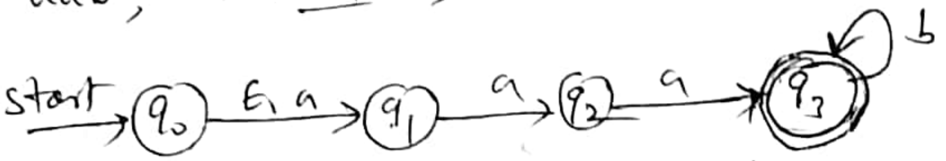
(5) Set of Final or accepting states denoted by F.

Problem: Design $\epsilon$-NDFSM for the following languages:

(1) $L = \{w \in \{a,b\}^* :$ w is made up of an optional a followed by aa followed by zero or more b's$\}$

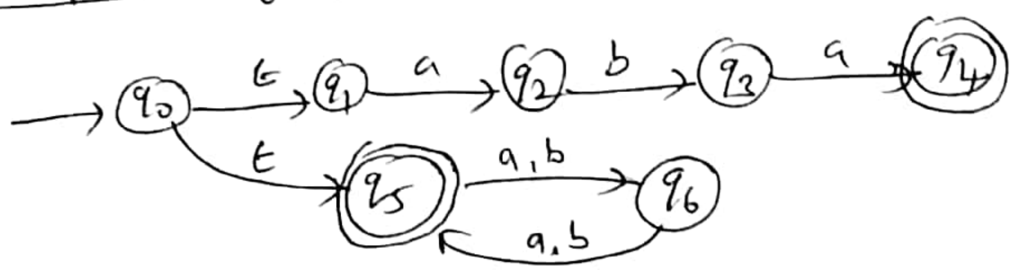
Sample strings w:

aa, aab, a aa bb, a aa bbb, .....




(2) $L = \{w \in \{a,b\}^* : w = aba$ or $|w|$ is even$\}$

Sample strings: aba, abba, aaba, babb, ....




(3) Let $\Sigma = \{a, b, c, d\}$

$L = \{w :$ there is a symbol $a_i$ in $\Sigma$ not appearing in string w.

Possible strings: bcbdbc, adcda, abdba, abcba,