Formal language and Automata theory

Computation :- The process of solving a problem to obtain a result is called computation.

-> The computation process can be represented by using Mathematical models

Types of mathematical models:-
The mathematical models are four types are there in given below
* finite automata
* push down automata
* linear Bounded automata
* Turing machine

finite automata:- finite automata is understanding only one language that language is called Regular language (RL) followed by with the help of regular grammer (RG)

Push down automata:- push down automata is understanding by particular language this language is known as context free language (CFL) is followed by with the understand by the also regular language

linear bounded automata:- linear bounded automata is understand for language is context sensitive language (CSL) and formed by the with the help of context sensitive grammer (CSG) and long with the understand by two more languages are RL and CFL language.

Turing Machine:- Turing machine is understand for language is recursive enumerable language (REL) and is followed by with the help of grammer is recursive enumerable grammer (REGL) and also along with the understand by three more language RL & CFL & CSL language

Relation btw automata:-
the relation btw above four language and grammers

$$RL \subseteq CFL \subseteq CSL \subseteq REL$$

$$RG \subseteq CFG \subseteq CSG \subseteq REG$$

Why study automata theory:-

* This theory is a fundamental course of comput.. science.
* Automata theory is the study of abstract mac.. and automata as well as the computational pro.. that can be solved using them.
* Automata theory will help you understand how people have thought about computer science a.. a science.
* Automata theory is mainly about
  1. What kind of things can you really compute mechanically
  2. How fast it take to do it (time complexity)
  3. How much space does it take to do it (space com)

Eg:- 1. Binary strings ends with 0?

1. 101011010 → accepted
2. 101000101 → Rejected.

Eg:- 2 Declaration statement in c language like int a,b,c;

The central concepts of Automata theory

Basic concepts:- 1. symbol
2. Alphabet
3. strings
4. languages.

1. Any formal language can be constructed by the basic concepts of automata theory.
2. Basic concepts of building blocks of automata theory.

1. symbol:- Symbol is an obj (or) a thing

Eg: a, b, c, d ----
0, 1, 2, 3 ----

#, *, +, -, @ . ...

symbols are used to form a string.

2. Alphabet :- It is a non empty and finite set of symbols

* It is denoted by Σ

eg:- $\Sigma = \{a, b, \ldots z\}$

$\Sigma = \{A, B, \ldots Z\}$

$\Sigma = \{0, 1, 2, \ldots 9\}$

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

$\Sigma = \{a, b, c, \ldots z, 0, 1, 2, \ldots 9\}$

3. string :- It is a sequence of symbols from Sigma

eg:- $s_1 = abc$

$s_2 = 010$

$s_3 = a,b,c$

$s_4 = 01234$

length of a string :- The no. of symbols appear in a given string s is called of length of s

* It is denoted by $|s|$

eg:- $s_1 = abc$ then $|s_1| = 3$

$s_2 = 010$ then $|s_2| = 3$

$s_3 = a,b,c$ then $|s_3| = 1$

$s_4 = 01234$ then $|s_4| = 5$

Language :- It is a collection of strings over sigma (Σ)

eg:- let $\Sigma = \{0, 1\}$

L_1 = set of strings have length of 2

$L_1 = \{00, 01, 10, 11\}$ → finite set

eg: L_2 = set of strings have length of 3

$L_2 = \{000, 001, 010, 011, 100, 101, 110, 111\}$ → finite set

eg: L_3 = set of strings ends with 0

$L_3 = \{000, 00, 0, 10, 1010, 11110 \ldots\}$ → infinite set

power of Σ :- $\Sigma\{0, 1\}$

Null string :- A string without symbol is a null string

It is denoted by ϵ

∴ length of null string $|\epsilon| = 0$

Σ^0 set of all strings length $0 = \{\epsilon\}$

Σ^1 set of all strings length $1 = \{0,1\}$

Σ^2 set of all strings length $2 = \{00, 01, 10, 11\}$

Σ^3 set of all strings length $3 = \{000, 001, 010, 011, 100, 110, 111\}$

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cdots\cdots \rightarrow$ star closure :- set of all strings including null string.

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots\cdots \rightarrow$ positive closure :- Excluding null string.

1. $\Sigma^* = \Sigma^0 \cup \Sigma^+$ $\qquad\qquad \Sigma^+ = \Sigma^* - \{\epsilon\}$

$= \epsilon \cup \Sigma^+$

$\boxed{\Sigma^* = \epsilon \cup \Sigma^+}$ $\qquad\qquad \boxed{\Sigma^+ = \Sigma^* - \epsilon}$

string operations :-

1. lengths of a string :- It means no. of symbols in the given string "s". It is denoted by $|s|$

Eg :- let s = "google" be a string over $\Sigma = \{a, b, c, \ldots z\}$

length $s \Rightarrow |s| = 7$

2. position of a symbol in string :- consider an input symbol "a" in the input string s the position "a" in s is "i" then it is denoted by $s_a(i)$

$\boxed{\begin{array}{l} s_a(i) = 1 \\ \quad\;\;\; = 0 \end{array}}$ if a is in i^{th} position of s
otherwise

Eg :- Consider an input symbol g in s then

$s_g(1) = 1$ $\qquad s_g(4) = 1$ $\qquad s_g(7) = 0$

$s_g(2) = 0$ $\qquad s_g(5) = 1$

$s_g(3) = 0$ $\qquad s_g(6) = 0$

3. concatenation :- It means combine two (or) more string into a single string.

$\rightarrow$ mathematically if s_1 & s_2 are two strings then the concatenation s' of s_1 & s_2 is given by

$s' = s_1 \cdot s_2$

$= \{xy \mid x \in s_1, y \in s_2\}$

$= \{yx \mid y \in s_1, x \in s_2\}$

Eg:- If s_1 = para and s_2 = graph then $s_1 o s_2$

$\therefore s_1 o s_2$ = paragraph

Eg:- If $\Sigma = \{a,b\}$, $s_1 = ab$

$s_2 = baa$

then $s' = s_1 o s_2 = abbaa$

Eg:- let $x = 0100101$ and $y = 1111$

$x o y = 01001011111$

Eg:- abba $o \epsilon$ = abba $|\epsilon| = 0$

Note:- Eg:- let z be any string then $z o \epsilon = \epsilon o z = z$
Identity rule

Note:- Associativity rule:- $a(bc) = ab(c)$

$a o (b o c) = (a o b) o c$

4. Reverse of a string:- Reverse of a string means, Reverse the symbols in a string.

→ It is denoted by s^R where s is given string

Eg:- 1. if s = mus then s^R = sum.

2. let s = bottle then s^R = elttob and $(s^R)^R$ = bottle

3. let $|s| = |s^R|$.

let x = klim and y = rettub then $(x o y)^R$

$x o y$ = klimrettub

$(x o y)^R$ = buttermilk.

$y^R o x^R = y^R$ = butter x^R = milk

$y^R o x^R$ = buttermilk $[\because (x o y)^R = y^R o x^R]$

Kleene closure (or) star closure :-

Kleene closure is introduced by kleene mathematician

It is a set which contains all strings including empty string (or) null string

It is denoted by s^*

It is used for regular expression

$s^* = \{s^0, s^1, s^2, s^3, s^4 \ldots\}$

$$\boxed{s^* = s^0 \cup s^1 \cup s^2 \cup s^3 \cup \ldots}$$

Eg:- if $s = \{a\}$ then find s^*

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

$s = \{a\}$

$S^0 = \{\epsilon\}$ $\quad$ $S^3 = \{aaa\}$

$S^1 = \{a\}$ $\quad$ $S^4 = \{aaaa\}$

$S^2 = \{aa\}$

$S^* = S^0 \cup S^1 \cup S^2 \cup S^3 \cup S^4 \cup \ldots$

$= \{\epsilon\} \cup \{a\} \cup \{aa\} \cup \{aaa\} \cup \{aaaa\} \cup \ldots$

$= \{\epsilon, a, aa, aaa, aaaa\}$

Eg:- if $S = a, b$ then find S^*

$S = \{a, b\}$

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

$S^0 = \{\epsilon\}$

$S^1 = \{a\}\{b\} = \{a, b\}$

$S^2 = \{aa, bb, ab, ba\}$

$S^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

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

$= \{\epsilon, a, b, aa, bb, ab, ba, aaa, aab, aba, abb, baa, bbb\}$

Eg:- if $S = \{cc, d\}$ then find S^*

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

$S^0 = \{\epsilon\}$

$S^1 = \{d\}$

$S^2 = \{cc, dd\}$

$S^3 = \{ccd, dcc, ddd\}$

$S^4 = \{cccc, ccdd, ddcc, dddd\}$

$\therefore S^* = S^0 \cup S^1 \cup S^2 \cup S^3 \cup \ldots$

$= \{\epsilon, d, cc, dd, ccd, dcc, ddd, cccc, ccdd, ddcc, dddd\}$

Positive closure:- It was introduced by Kleen mathematician

-> It is a set which contains all strings excluding null (or) empty string.

-> Denoted by S^+

-> used for regular expression

$$\boxed{\therefore S^+ = S^1 \cup S^2 \cup S^3 \cup \ldots}$$

Eg:- (1) if $s = \{a\}$ then s^+

$s^+ = s^1 \cup s^2 \cup s^3 \cup s^4 \cup ---$

$s^1 = \{a\}$ $\quad$ $s^4 = \{aaaa\}$

$s^2 = \{aa\}$

$s^3 = \{aaa\}$

$s^+ = s^1 \cup s^2 \cup s^3 \cup s^4 \cup ----$

$= \{a, aa, aaa, aaaa\}$

2) if $s = \{a, ba\}$ then find s^+

$s^1 = \{a\}$

$s^2 = \{aa, ba\}$

$s^3 = \{aaa, aba, baa\}$

$s^4 = \{aaaa, aaba, baaa, baba\}$

$s^+ = s^1 \cup s^2 \cup s^3 \cup s^4 ----$

$= \{a, aa, ba, aaa, baa, aba, aaaa, aaba, baaa, baba\}$

Note :- Relation ship b/w s^* and s^+

$s^* = s^0 \cup s^+$ $\qquad$ $s^+ = s^* - s^0$

$\boxed{s^* = \epsilon \cup s^+}$ $\qquad$ $\boxed{s^+ = s^* - \epsilon}$

Parts of a string :- 1. prefix of a string

2. Suffix of a string

3. proper prefix of a string

4. proper suffix of a string.

1) prefix of a string :- The number of leading symbols of given string.

=> let 'x' be a string then prefix of 'x' is denoted by prefix(x).

eg:- if x=abc is a string then prefix of x is

prefix(x) = {ϵ, a, ab, abc}

2) Suffix of a string :- The number of trailing symbols in a given string

→ It is denoted by suffix(x)

eg:- if x=abc is a string then suffix of x is

suffix(x) = {ϵ, c, bc, abc}

3) proper prefix of a string :- The no.of leading symbols except the given string.

-> It is denoted by proper prefix (x)

Eg:- if x=abc is a string then proper prefix (x)

proper prefix (x) = {ε, a, ab}

4) proper suffix of a string:- The no.of trailing symbols except the given string.

-> It is denoted by proper suffix (x)

Eg: If x = abc is a string then proper suffix (x)

proper suffix (x) = {ε, c, bc}

language:-

* Introduction * operations of language

Introduction:- A language is a finite and non empty set of strings. It is denoted by L

-> All these strings are formed from the alphabet Σ

language notation L(M):- is a language defined by machine M that accepts a set of strings.

-> L(G) is a language defined by the Grammar "G" that recognize a set of strings.

-> L(R) is a language defined by the regular expression that represent a set of strings.

Eg:-

1) let Σ = {z} then the language of all possible strings is given by Σ = {z}

$L = \{ε, z, zz, zzz, zzzz\}$

2) The language of all possible of even length strings over Σ = {z}

$L = \{z^0, z^2, z^4, z^6, z^8 ----\}$

$L = \{z^{2n} / n \geq 0\}$

Eg:- The language of all possible strings of odd length over. Σ = {z}

$L = \{z^1 z^3 z^5 z^7 z^9 ----\}$

$L = \{ 7^{2n-1} \mid n \geq 1 \}$

Operations on language:-

1. Union
2. Intersection
3. Complementation
4. Symmetric difference
5. Reversal of language
6. Polindrome language
7. Kleene closure (or) star closure of language
8. Demorgan's law

1) Union:- It is a simple operation on two languages.
-> let L_1 & L_2 be two languages then the union is defined by $L_1 \cup L_2$.
-> mathematically the union of two languages is defined as $L_1 \cup L_2 = \{x \mid x \in L_1 \text{ or } x \in L_2\}$

Eg:- let $L_1 = \{0, 01, 011\}$ and $L_2 = \{\epsilon, 001, 1^5\}$ then $L_1 \cup L_2 =$
$\{\epsilon, 0, 01, 011, 001, 1^5\}$

2) let $L_i = 2^i$ then $\bigcup_{i=0}^{7} L_i =$

$\bigcup_{i=0}^{7} L_i = L_0 \cup L_1 \cup L_2 \cup L_3 \cup L_4 \cup L_5 \cup L_6 \cup L_7$
$= 2^0 \cup 2^1 \cup 2^2 \cup 2^3 \cup 2^4 \cup 2^5 \cup 2^6 \cup 2^7$
$= \{\epsilon\} \cup \{2\} \cup \{22\} \cup \{222\} \cup \{2222\} \cup \{22222\} \cup \{222222\} \cup \{2222222\}$,

$\{\epsilon, 2, 22, 222, 2222, 22222, 222222, 2222222\}$

2) Intersection:- It is a simple operation on two languages and L_1 & L_2 be two languages and their intersection is denoted by $L_1 \cap L_2$
-> mathematically intersection of L_1 & L_2 is defined as
$L_1 \cap L_2 = \{x \mid x \in L_1 \text{ and } x \in L_2\}$

let $L_1 = \{\epsilon, 00, 0000, 000000, ---\}$ and
$L_2 = \{\epsilon, 0, 000, 00000, ---\}$ then $L_1 \cap L_2$
$L_1 \cap L_2 = \{\epsilon\}$

Let $L_1 = \{\epsilon\}$ and $L_2 = \{\phi\}$ $\therefore L_1 \cap L_2 = \phi$

3) Complementation :-

It is a simple operation performed on single langu..

-> let L be a language over Σ then the complementat..
of L is denoted by $\bar{L}$ (or) L^c therefore $\bar{L}$ or L^c =

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

-> Mathematically the complementation of L is defi..
as $\bar{L} = \{x \mid x \in \Sigma^* \text{ and } x \notin L\}$

Eg:- if $\Sigma = \{a,b\}$ and $L = \{a, b, aa\}$ then L^c = ?

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

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

$\Sigma^1 = \{a\} \{b\}$

$\Sigma^2 = \{aa, ab, ba, bb\}$

$\Sigma^3 = \{aaa, aab, baa, aba, bba, bab, bbb, abb\}$

$\therefore L^c = \Sigma^* - L$

$= \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, baa, aba, bba, bab, bbb, abb\} - \{a, b, aa\}$

$= \{\epsilon, ab, ba, bb, aaa, aab, baa, aba, bba, bab, bbb, abb\}$

2) let $L = \{\epsilon, 1, 11, 111, 1111 ---\}$ over $\Sigma = \{0,1\}$ then find $\bar{L}$?

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

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

$\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, ---\} - \{\epsilon, 1, 11, 111, 1111 ---\}$

$= \{0, 00, 01, 10, 000, 001, 010, 011, 100, 101, 110 ---\}$

Symmetric difference :-

It is a simple operation on two languages.

→ let L_1 & L_2 be two languages then the symmetric operation on L_1 & L_2 is defined as

$$\boxed{L_1 \oplus L_2 = (L_1 \cup L_2) - (L_1 \cap L_2)}$$

→ Here x is in L_1 or L_2 but not in both.

Eg:- let L be a language over Σ

1) $L \oplus \phi = (L \cup \phi) - (L \cap \phi)$
$= L - \phi$
$= L$

2) $L \oplus L = (L \cup L) - (L \cap L)$
$= L - L$
$= \phi$

3) $L \oplus \Sigma^* = (L \cup \Sigma^*) - (L \cap \Sigma^*)$
$= \Sigma^* - L$
$= \Sigma$

4) $L \oplus \Sigma = (L \cup \Sigma) - (L \cap \Sigma)$
$= \Sigma^* - \phi$
$= \Sigma^*$

Eg:- let $L_1 = \{00, 0000, 000000, \ldots\}$ $L_2 = \{11, 111, 11111, \ldots\}$ then

$L_1 \oplus L_2 = (L_1 \cup L_2) - (L_1 \cap L_2)$
$= \{00, 0000, 000000 \ldots, 11, 111, 11111, \ldots\} - \{\phi\}$
$= \{00, 0000, 000000, \ldots, 11, 111, 11111, \ldots\}$

Eg:- If $\Sigma = \{0, 1\}$ then $\Sigma^{\leq 2} \oplus \{\epsilon, 0, 00, 101, \ldots\}$

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

$\Sigma^{\leq 2} \oplus \{\epsilon, 0, 00, 101\}$
$= (\Sigma^{\leq 2} \cup \{\epsilon, 0, 00, 101, \ldots\}) - (\Sigma^{\leq 2} \cap \{\epsilon, 0, 00, 101, \ldots\})$

$= \{\epsilon, 0, 1, 00, 01, 10, 11\} \cup \{\epsilon, 0, 00, 101, \cdots\} - \{\epsilon, 0, 1, 00, 01, 10, 11\}$
$\cap \{\epsilon, 0, 00, 101\}$.

concatenation of language:-

concatenation of language used to combine two or more languages into a single language. It is denoted by $L_1 o L_2$ or $L_1 L_2$.

Mathematically it is denoted by

$$L_1 o L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$$

Eg:- 1) Let $L_1 = \{bc, bcc, cc\}$ and $L_2 = \{cc, cccc\}$ then
i) $L_1 o L_2 = ?$ ii) $L_2 o L_1 = ?$

$L_1 o L_2 = \{bc, bcc, cc\} o \{cc, cccc\}$
$= \{bccc, bccccc, bcccc, bcccccc, cccc, cccccc\}$

$L_2 o L_1 = \{cc, cccc\} o \{bc, bcc, cc\}$
$= \{ccbc, ccbcc, cccc, ccccbc, ccccbcc, cccccc\}$

2) Let $L_1 = \{0,1\}^*$ and $L_2 = \{0,1\}$ then $L_1 o L_2 = ?$

$L_1 = \{0,1\}^*$
$L_1 = L_1^0 \cup L_1^1 \cup L_1^2 \cup L_1^3 \cup L_1^4 \cdots$
$L_1^0 = \{\epsilon\}$
$L_1^1 = \{0,1\}$
$L_1^2 = \{00, 01, 10, 11\}$
$L_1^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$
$L_1 = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \cdots\}$
$L_2 = \{0,1\}$
$L_1 o L_2 = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \cdots\}$
$\{0,1\}$.

$= \{0, 00, 10, 000, 010, 110, 000, 0010, 0100, 0110, 1000, 1010, 1100, 1110,$
$1, 01, 11, 001, 011, 101, 111, 0001, 0011, 0101, 0111, 1001, 1101, 1111\}$

3) for any language $L \circ \epsilon = \epsilon \circ L = L$

4) for any language $L \circ \phi = \phi \circ L = \phi$

Reversal of language:-

Language = set of strings.

It is similar to Reverse of strings.

It means Reverse the all strings in that language

It can be denoted by "L^R"

mathematically it can be defined as $L^R = \{w^R / w \in L\}$

eg:- 1) If $L = \{a, b, a_2b_2, a_3b_3\}$ then L^R

$L^R = \{a, b, b_2a_2, b_3a_3\}$

2) If $L = \{0, 01, 011\}$ then L^R

$L^R = \{0, 10, 110\}$

3) If $L = \{0^i 1^{i+1} / i \geq 0\}$ then L^R

$L^R = \{1^{i+1} 0^i / i \geq 0\}$

Kleene closure of a language:-

It is a set of all strings including null string (or) Empty string.

It is denoted by L^*.

mathematically it is defined as $L^* = \cup L^i, i \geq 0$

Here, 'i' is 0 number of x.

Positive closure of a language:-

It is a set of all strings excluding Null string (or) Empty string.

It is denoted by L^+.

Mathematically it is defined as $L^+ = L^1 \cup L^2 \cup L^3 \cup$

$L^+ = \cup x^i, i \geq 1$

Eg:- If $L = \{\epsilon, z, zz, zzz, ---\}$ over $\Sigma = \{z\}$ then $L^* = ?$

$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup --$

$L^0 = \{\epsilon\}$

$L^1 = \{z\}$

$L^2 = \{zz\}$

$L^3 = \{zzz\}$

$L^* = \{\epsilon, z, zz, zzz, ---\}$

2) If $L = \{\epsilon, 0, 1, 00, 01, 10, 11, --\}$ over $\Sigma = \{0, 1\}$ then $L^* = ?$

$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup ---$

$L^0 = \{\epsilon\}$

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

$L^2 = \{00, 01, 10, 11\}$

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

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

eg:-3) If $\{000\}^* = ?$

$\{000\}^* = \{000\}^0 \cup \{000\}^1 \cup \{000\}^2 \cup \{000\}^3 \cup \cdots$

$= \{\epsilon\} \cup \{000\} \cup \{000000\} \cup \{000\,000\,000\} \cup \cdots$

$= \{\epsilon, 000, 000000, 000000000\}$

4) $\phi^* = ?$

$\{\phi\}^* = \{\phi\}^0 \cup \{\phi\}^1 \cup \{\phi\}^2 \cup \{\phi\}^3 \cup \cdots$

$= \{\epsilon\} \cup \{\ \} \cup \{\ \} \cup \{\ \} \cup \cdots$

$= \{\epsilon\}$

5) If $L = \{\epsilon\}$ then $L^* = ?$

$L^* = L^0 \cup L^1 \cup L^2 \cup L^3 \cup \cdots$

$L^0 = \{\epsilon\}$

$L^1 = \{\epsilon\}^1$

$L^2 = \{\epsilon\}^2$

$L^3 = \{\epsilon\}^3$

$\therefore L^* = \{\epsilon\} \cup \{\epsilon\}^1 \cup \{\epsilon\}^2 \cup \{\epsilon\}^3$

$= \{\epsilon\}$

Demorgan's laws :-

Demorgan's laws used to express the intersection of languages interms of union and different languages.

i) $L_1-(L_2 \cup L_3) = (L_1-L_2) \cap (L_1-L_3)$

ii) $L_1-(L_2 \cap L_3) = (L_1-L_2) \cup (L_1-L_3)$

iii) $(L_1 \cup L_2)' = L_1' \cap L_2'$

iv) $(L_1 \cap L_2)' = L_1' \cup L_2'$

v) $L_1 \cap L_2 = (L_1' \cup L_2')'$

Finite Automata :-

* Introduction
* Components of FA
* elements of FA
* Representation of FA.
* Examples.

Introduction :-

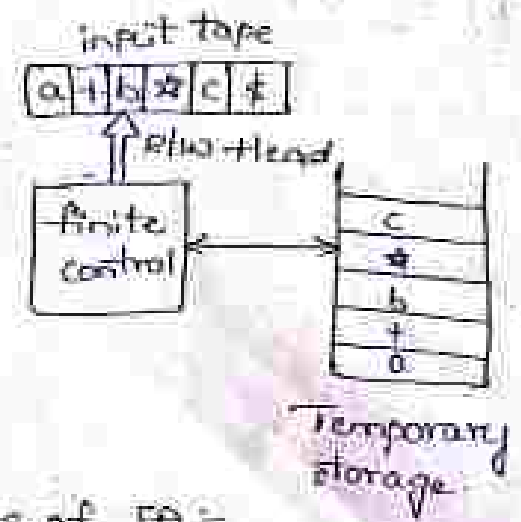
It is a self operating machine.

It is a system which obtains, transforms, transmits and uses information to perform its functions without direct participation of humans.

Finite automata is used in mathematical analysis

Application of finite Automata is lexical Analysis phases in compiler design.

Model of finite automata :-



Components of FA :-

1) Input tape 2) R/W Head 3) Temporary storage

4) Finite control.

1) Input tape :-

It is a memory storage used to store the input data memory area is divided into number of cells

Each cell can hold only one input symbol at a time

- The input string ends with end marker

2) R/W Head :-

It is used by the finite control to read the input data from left side to right side in the input tape.

R/W head can move from left to right and reads only one input symbol at a time

3) Finite control unit :-

* It controls the entire process of a system (or) machine.
* Finite control reads the input data from the input buffer tape by moving read or write head from left to right
* It stores the reading data in temporary storage
* It stops the reading process when the read or write head reaches to end marker in the input tape

Elements of finite Automata :-

1. state 2. Transitions 3. Transition diagram / state diagram
4. Transition table

1) state :- It is a behaviour that produce an action

* It is a location in input buffer
* the finite control reads the data from one state to another state in the buffer
* states are classified into four types
  i) initial state (or) starting state
  ii) Final state (or) Acceptance state
  iii) Intermediate state
  iv) Invalid state (or) Dead (or) Trap state

Initial state (or) starting state :-

The finite control can starts its reading process from a state is called initial state

2. <u>Finite state or acceptance state</u>:-

- Finite control can stops its reading process after completion or end of given string then it reaches a state that state is called final state

3. <u>Intermediate state</u>:-

The states between starting state and final state are called Internal (or) Intermediate state

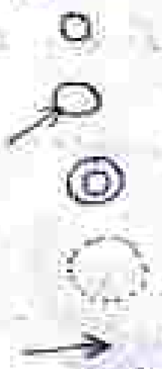
4. <u>Invalid state (or) Dead (or) Trap state</u>: It is a invalid state when the finite control reads the input data from dead state then it always goes to dead state

2. Transitions:-

* It means moving one place to another place
* It is defined by Transition function.
* Transition function is denoted by <u>δ</u>.

3. Transition Diagram:-

* It is a graphical representation of finite automata.
* It is a directed graph.
* It is constructed by using the following symbols (or) Notations

<u>symbol</u>	<u>Meaning</u>
	state
	initial state
	final state
	dead state
	Transition

<u>Ex</u>:-

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

$\delta(q_2, b) = q_2$

4. Transition tables :-

- It is a tabular representation of finite automata.
- It is combination of rows and columns. Here rows represent states, columns represents input symbols. The entry in between row and column is always states.

eg:- 

state	input symbols	
	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_2

** Representation of finite automata :-

Mathematically a finite automata is a 5-tuple machine like $M = (Q, \Sigma, \delta, q_0, F)$ where,

$Q \rightarrow$ set of states

Q is a finite and non-empty set of states.

$\Sigma \rightarrow$ It is finite and non-empty set of input symbols

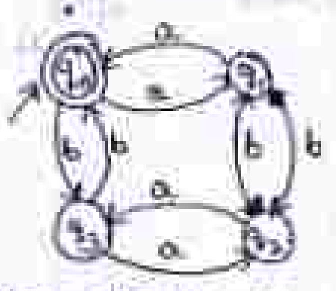
$\delta \rightarrow$ It is a transition function which is defined as $\delta : Q \times \Sigma \rightarrow Q$

$q_0 \rightarrow$ It is a initial state or starting state and $q_0 \in Q$

$F \rightarrow$ It is a finite set of final states and $F \subseteq Q$

Note :- * finite automata contains one initial state

* finite automata contains one or more final states

Ex :- 

mathematically it represented as

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

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

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

δ is a transition function which is defined as δ:

$Q \times \Sigma = Q$

Transition table:

δ: state	input a	b
$\rightarrow q_0$	q_1	q_3
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_2	q_0

$q_0 = \{q_0\}$

$F = \{q_0\}$

** Acceptance of a string by FA:-

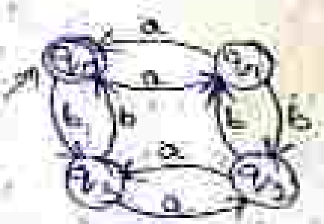
Let w be a given string and finite automata M contains Q states like $Q = \{q_0, q_1, q_2, ---- q_f\}$ where q_0 is the initial state q_f is the final state. The string w is accepted by machine M. If and only if $\delta(q_0, w) = q_f$.

ex:- check whether the following strings are accepted or not by the given finite automata.

i) aabb ii) ababa iii) aabbaa iv) abababb

sol:-

The given finite automata



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

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

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

δ: state	input a	b
$\rightarrow q_0$	q_1	q_3

q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_2	q_0

$q_0 = \{q_0\}$

$F = \{q_0\}$

i) $w = aabb$

$$\delta(q_0, aabb) = \delta(q_1, abb)$$
$$= \delta(q_0, bb)$$
$$= \delta(q_2, b)$$
$$= q_0 \in F$$

$\therefore$ The given string $w = aabb$ is accepted by F.A M.

ii) $w = ababa$

$$\delta(q_0, ababa) = \delta(q_1, baba)$$
$$= \delta(q_2, aba)$$
$$= \delta(q_3, ba)$$
$$= \delta(q_0, a)$$
$$= q_1$$

$\therefore$ The given string $w = ababa$ is not accepted by F.A

iii) $w = aabbaa$

$$\delta(q_0, aabbaa) = \delta(q_1, abbaa)$$
$$= \delta(q_0, bbaa)$$
$$= \delta(q_2, baa)$$
$$= \delta(q_0, aa)$$
$$= \delta(q_1, a)$$
$$= q_0 \in F$$

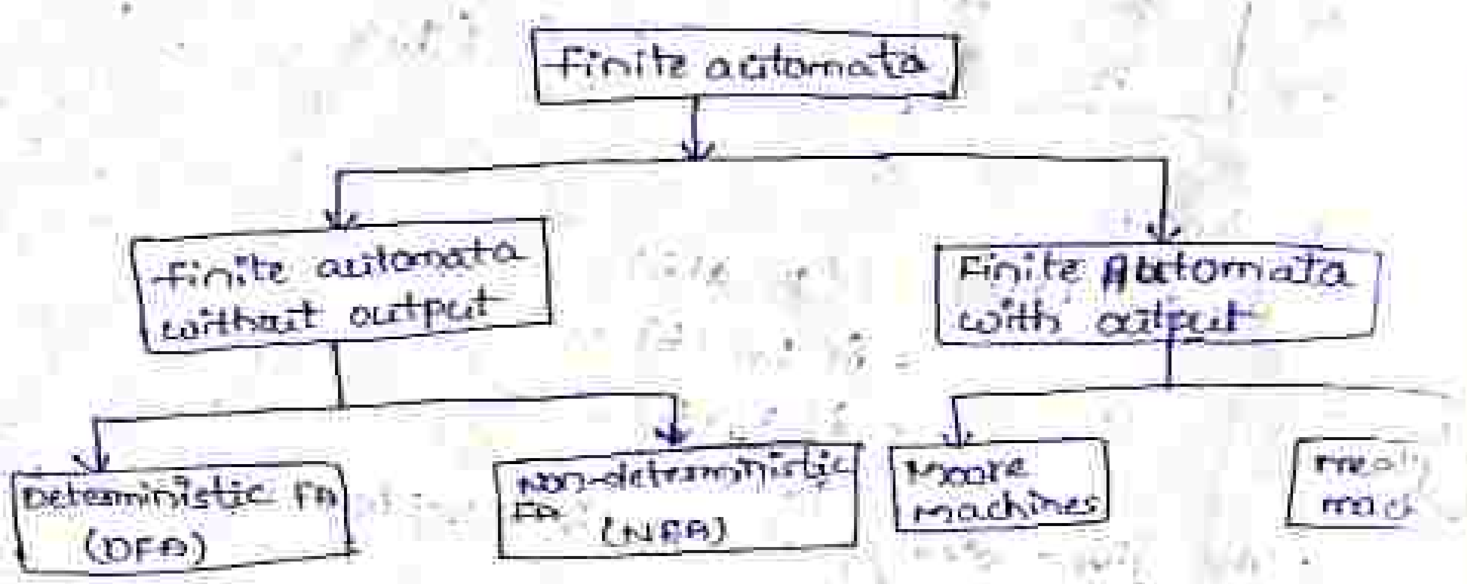
$\therefore$ The given string $w = aabbaa$ is accepted by F.A M.

iv) $w = abababb$

$$\delta(q_0, abababb) = \delta(q_1, bababb)$$
$$= \delta(q_2, ababb)$$
$$= \delta(q_3, babb)$$
$$= \delta(q_0, abb)$$
$$= \delta(q_1, bb)$$
$$= \delta(q_2, b)$$
$$= q_1$$

$\therefore$ The given string $w = abababb$ is not accepted by F.A

Types of FA:-



Deterministic FA:-

* Introduction *Representation *language Accep by DFA

* Acceptance of DFA string by DFA

* Design of DFA.

Introduction:-

We can determine exactly what is the next state by reading a particular input symbol from a particular state then that FA is called DFA

Definition:-

A DFA is a finite state machine where for each pair of current state and current input symbol there is a unique next state.

Representation of DFA:-

mathematically DFA is five Tuples like

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

$Q \rightarrow$ finite and non-Empty sets of states

$\Sigma \rightarrow$ finite and non-Empty sets of input symbol

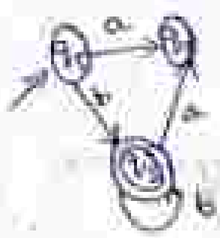
$\delta \rightarrow$ is a Transition function is defined as

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

$q_0 \rightarrow$ is the initial state

$F \rightarrow$ is the final state

Example of DFA:-



specification of DFA:-

1) Transition diagram   2) Transition table.

Acceptance of a string by DFA

consider the deterministic finite Automata 'M' and the string 'w' over input Alphabet Σ. Now, the string 'w' is Accepted by 'M' if and only if $L(M) = \{ w \mid w \in \Sigma, \delta(q_0, w) = q_f \}$

methods for check whether the given string is accepted or not by DFA.

There are 3 methods

1) Using sequence diagram.
2) Using extended Transition function.
3) Using V dash-function.

eg:- 1) consider a DFA now check whether the following strings are accepted or not by the DFA using 1) sequence diagram 2) Transition function 3) V dash function

1) 100 2) 1101 3) 100100

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

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

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

δ:

states	inputs 0	inputs 1
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

$q_0 = q_0$

$F = \{q_2\}$

i. $w = 100$

1) using sequence diagram

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

∴ The given string $w=100$ is accepted by the DFA

2) Using extended transition function:-

$w=100$

$\delta(q_0, w) = \delta(q_0, 100)$

$= \delta(q_0, 00)$ ∴ the given string is accepted by FA

$= \delta(q_1, 0)$

$= q_2 \in F$

3) Using ⊢ dash function:-

$w=100$

$\vdash (q_0, 100) = \vdash M (q_0, 00)$

$= \vdash M (q_1, 0)$

$= q_2 \in F$

∴ The given string is accepted by FA.

** Design of DFA:-

procedure:- 1) understanding the language which is for designing a DFA.

2) Determine minimum length string in the language.

3) Draw the DFA for minimum length string.

4) Determine initial, Intermediate, dead and final states of DFA

5) Apply each input symbol on every state of DFA.

Eg:-

1) Design a DFA for the language which consists of set of all strings of 0's over $\Sigma = \{0\}$

Let 'M' be a DFA with 5 tuples like

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

Given input $\Sigma = \{0\}$

$\therefore L(M) = \{\epsilon, 0, 00, 000 \ldots\}$

Minimum string is ϵ

The next minimum string is '0'.

∴ DFA 

$Q = \{q_0\}$

$\Sigma = \{0\}$

δ :

state	input 0
→q_0	q_0

$q_0 = q_0$

$F = \{q_0\}$

2) Design a DFA for the language consist of set of all strings over accept Empty string over $\Sigma = \{0\}$

Given input $\Sigma = \{0\}$

let M be a DFA like $M = (Q, \Sigma, \delta, q_0, F)$

∴ $L(M) = \{0, 00, 000, 0000 \cdots\}$

Minimum string is '0'.

∴ DFA

∴ $Q = \{q_0, q_1\}$

$\Sigma = \{0\}$

δ :

state	input 0
→q_0	q_1
q_1	q_1

$q_0 = q_0$

$F = \{q_1\}$

3) construct a DFA that accepts a language over $\{0,1\}^*$

let 'M' be a DFA like

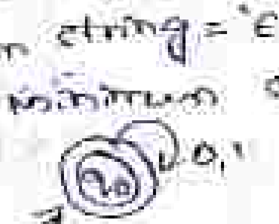
$M = (Q, \Sigma, \delta, q_0, F)$

The given alphabet $\Sigma = \{0,1\}^*$

∴ $L(M) = \{\epsilon, 0, 1, 00, 01, 10, 11\}$

minimum string = 'ϵ'

The Next minimum string is '0 or 1'

∴ DFA 

$q_0 = q_0$

$F = \{q_0\}$

$Q = \{q_0\}$

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

δ :

state	input 0	input 1
→q_0	q_0	q_0

4) construct a DFA that accepts a language over set of $\{0,1\}^+$

let M be a DFA like

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

The given $\Sigma = \{0,1\}^+$

$\therefore L(M) = \{0, 1, 00, 01, 10, 11 - - -\}$

The minimum string is 0 or 1.

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

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

δ:

state	input 0	1
$\rightarrow q_0$	q_1	q_1
$\rightarrow$ (q1)	q_1	q_1

$q_0 = q_0$

$F = \{q_1\}$

5) construct a DFA that accepts a language which contains set of all strings with even number of a's over $\Sigma = \{a\}$

$\Rightarrow$ let M be a DFA like

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

The given input $\Sigma = \{a\}$

$\therefore L(M) = \{a^0, a^2, a^4, a^6, a^8, - - \}$

$= \{\epsilon, aa, aaaa, aaaaaa, - - - \}$

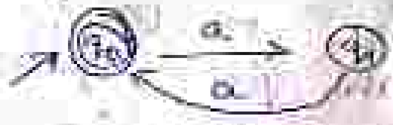
minimum string is 'ϵ'

Next minimum string = aa

$\therefore$ DFA

$Q = \{q_0, q_1\}$

$\Sigma = \{a\}$



δ:

state	input a
$\rightarrow$ (q0)	q_1
q_1	q_1

$q_0 = q_0$

$F = \{q_1\}$

6) Construct a DFA that accepts a language which contains set of all strings with odd number of b's over $\Sigma = \{b\}$

$\Rightarrow$ let M be a DFA like

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

$\therefore$ The given $\Sigma = \{b\}$

$\therefore L(M) = \{b^1, b^3, b^5, b^7, - - - \}$

$= \{ b, bbb, bbbbb, bbbbbbb, \ldots \}$

minimum string is 'b'

$Q = \{q_0, q_1\}$

$\Sigma = \{b\}$

δ:

state	input b
$\rightarrow q_0$	q_1
q_1	q_0

$q_0 = q_0$

$F = \{q_1\}$

construct a DFA for a language contains the following over $\Sigma = \{0,1\}$

i) set of all strings with even no.of 0's and even no.of 1's.

ii) set of all strings with even no.of 1's and odd no.of 0's.

iii) set of all strings with odd no.of 1's and even no.of 0's.

iv) set of all strings with odd no.of 1's and odd no.of 0's.

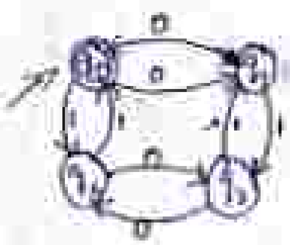
sol: $\Sigma = \{0,1\}$

1 - odd

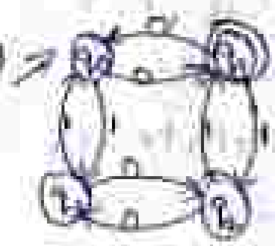
0 - even

1	0	final state(s)
0	0 = 0	q_0
0	1 = 1	q_1
1	0 = 2	q_2
1	1 = 3	q_3

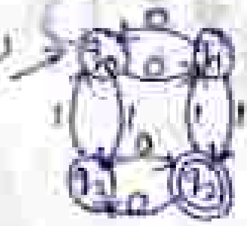
i)



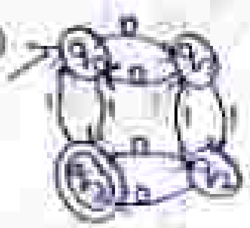
ii)



iii)



iv)



1) Design a DFA that accepts a language containing of all strings which are divisible by 3, where string is created as binary string.

sol:- $\Sigma = \{0,1\}$

$L(M) = \{$All strings divisible by 3$\}$

$\therefore$ The possible remainders are

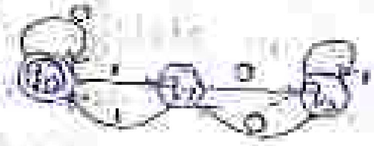
0, 1, 2

$\therefore$ The states are q_0, q_1, q_2

δ: initial state $= q_0$

final state $= q_0$

	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2



2) Construct a DFA for the language $L = \{W\}$ ~~such that~~ where string is created as ternary number.

sol:- $\Sigma = \{0,1,2\}$

$\therefore L(M) = \{$All strings divisible by 5$\}$

$\therefore$ The possible remainders are

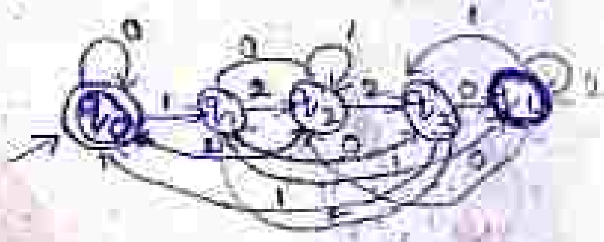
0, 1, 2, 3, 4.

$\therefore$ The states are q_0, q_1, q_2, q_3, q_4

δ: initial state $= q_0$

final state $= q_0$

δ:	0	1	2
$\rightarrow q_0$	q_0	q_1	q_2
q_1	q_3	q_4	q_0
q_2	q_1	q_2	q_3
q_3	q_4	q_0	q_1
q_4	q_2	q_3	q_4



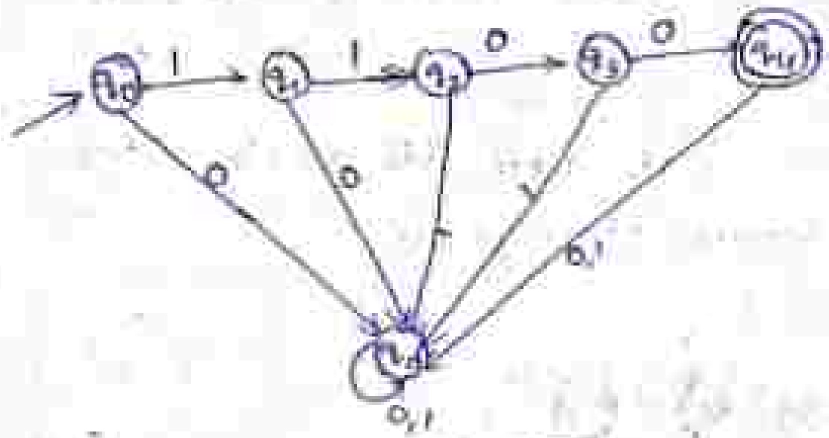
model 3

1) Construct a DFA that accepts a language which contains the string 1100 only over $\Sigma = \{0,1\}$

⇒ $\Sigma = \{0,1\}$

$L(M) = \{1100\}$

The minimum string = 1100



q_d is the invalid state

2) Design a DFA that accepts set of all strings that contains 0's and 1's and ends with 00 over $\Sigma = \{0,1\}$

Sol: Let 'M' be a DFA like

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

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

The strings ends with 00

$(0+1)^* 00$

$L(M) = \{00, 000, 100, 0000, 1100, 0100, 1000 - - -\}$

minimum string = 00

δ:

	0	1
$\to q_0$	q_1	q_0
q_1	q_2	q_0
(q_2)	q_2	q_0

3) Design a DFA that accepts a language of set of strings which are starts with 'a' and ends with 'b' over $\Sigma = \{a,b\}$

⇒ let 'M' be a DFA like

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

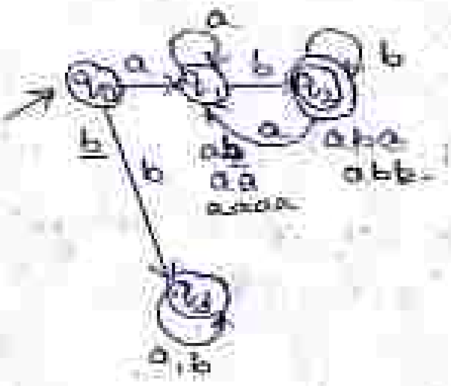
$\Sigma = \{a,b\}$

The strings starts with 'a' and ends with 'b'

$a\,(a+b)^*\,b$

$L(M) = \{ab, aab, abb, aaab, abbb, ---\}$

minimum string = ab



δ:

	a	b
→ q_0	q_1	q_d
q_1	q_1	q_2
(q_2)	q_1	q_2

q_d this is a dead state

4) Design a DFA that accept a language of all strings which are starts with ab over $\Sigma = \{a,b\}$

sol: Let 'M' be a DFA like

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

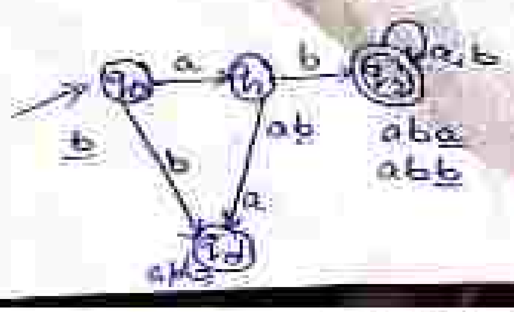
$\Sigma = \{a,b\}$

The strings starts with 'ab'

$ab\,(a+b)^*$

$L(M) = \{ab, aba, abb, abaa, abbb, ---\}$

minimum string = ab.



δ:

	a	b
→ q_0	q_1	q_d
q_1	q_d	q_2
(q_2)	q_2	q_2
q_d		

Model-1

1) Design a DFA that accepts a language of set of all strings of a's and b's which contains abb as a substring over $\Sigma = \{a,b\}$

=) Let 'M' be a DFA like

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

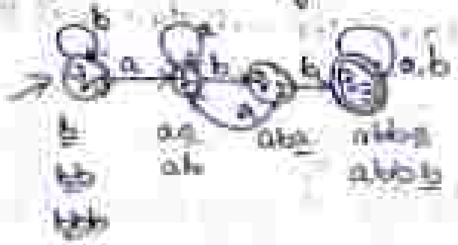
$\Sigma = \{a,b\}$

The strings with substrings as abb

$\underline{(a+b)^*}\ abb\ \underline{(a+b)^*}$

L(M) = { abb, aabbb, babb, abba, abbb, aaabb, bbabb, abbaa, abbbb, ----}

Minimum string = abb.



δ:

	a	b	b
$\rightarrow q_0$	q_1	q_0	q_0
q_1	q_1	q_2	q_2
q_2	q_1	q_3	q_3
q_3	q_3	q_3	q_3

2) Design a DFA over $\Sigma = \{a,b\}$ that contains set of strings of a's and b's <u>Except</u> those containing substring bba

=) Let 'M' be a DFA like

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

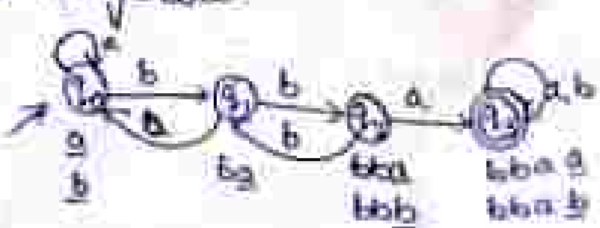
$\Sigma = \{a,b\}$

The strings with does not contains the substrings as bba

$\underline{(a+b)^*}\ bba\ \underline{(a+b)^*}$

L(M) = { bba, abba, bbba, bbaa, abbab, --}

Minimum string = bba



δ:

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

interchange

final state → non-final state

non-final state → final state

The required DFA is

3) Design a DFA over $\Sigma = \{0,1\}$ that contains set of string are 0's and 1's and those strings does not containing the substring 001

⇒ Let M be a DFA like

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

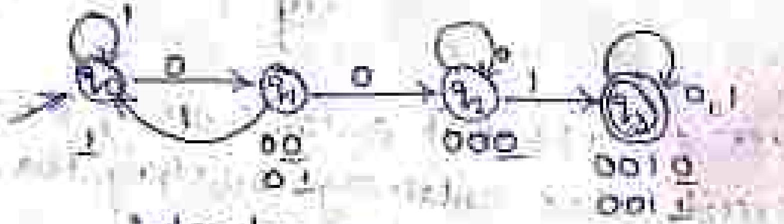
$\Sigma = \{0,1\}$

Each string does not contains the substring is 001

$\underline{(0+1)^*}\ 001\ \underline{(0+1)^*}$

$L(M) = \{001, 0001, 1001, 0010, 0011 \cdots\}$

minimum string = 001



δ:

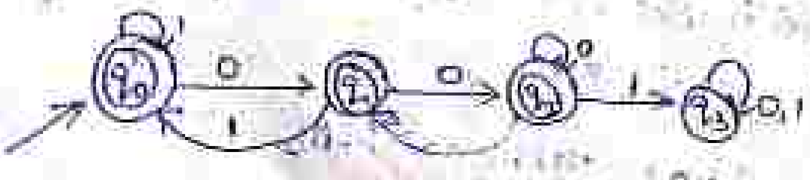
	0	0	1
$\rightarrow q_0$	q_1	q_1	q_0
q_1	q_2	q_2	q_0
q_2	q_2	q_2	q_3
(q_3)	q_3	q_3	q_3

interchange

final state → nonfinal state

nonfinal state → final state

The required DFA is



$\{w \in \{a,b\}^* /$ w do not bb or ...

Non-Deterministic finite Automata

Introduction:

We cannot determine the next state exactly after reading an inputsymbol from a particular state then that FA is called NFA.

Definition:-

NFA is a finite state machine wherever each pair of current state and particular input symbol it has more than one next state

Elements:-

1) states 2) input symbols 3) Initial state 4) final state

5) Transitions.

Representation of NFA:-

Mathematically NFA is a five tuple like $M=(Q,\Sigma,\delta,q_0,F)$

where Q = finite and non empty set of states

Σ = finite and non empty set of input symbols (or) input alphabets.

δ = It is a transition function which is defined as

$\boxed{Q \times \Sigma \longrightarrow 2^Q}$

q_0 = initial state it must be belongs to Q

F = finate state and $F \subseteq Q$

Description of NFA:-

It is of two ways i) Transition diagram ii) Transition table

extended transition function:-

1. $\delta(q,\epsilon) = q$ 2. $\delta(q_i,a) = q_j$ where $q_i, q_j \in Q$

3. $\delta(q,xa) = \delta(\delta(q,x),a)$

language accepted by NFA:-

Mathematically language accepted by NFA is defined as

$L(M) = \{w \mid \delta(q_0,w) = q_f \in F\}$

Design of NFA:—

1) procedure: understanding the language which is for designing a NFA.
2) Determine minimum length string in the language
3) Draw the NFA for minimum length string.
4) Determine initial, Intermediate, dead and final states of NFA.
5) Apply the each input symbol on initial and final states of NFA

1) Design a NFA that accepts set of all strings over Σ that have atleast two consecutives 0's (or) 1's.

⇒ Let 'M' be a NFA like

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

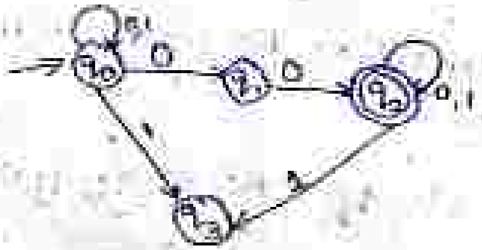
$\Sigma = \{0, 1\}$

Each string has atleast two consecutives 0's or 1's

$\underline{(0+1)}^* (00+11) \underline{(0+1)}^*$

$L(M) = \{00, 11, 000, 011, 100, 111, \ldots\}$

minimum string = 00, 0011



δ:

states	input 0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0, q_3\}$
q_1	q_2	–
(q_2)	q_2	q_2
q_3	–	q_2

2) Design an NFA to accept set of all strings starting with a followed by a or b and ending with a or any no. of b's over $\Sigma = \{a, b\}$

⇒ Let 'M' be NFA like

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

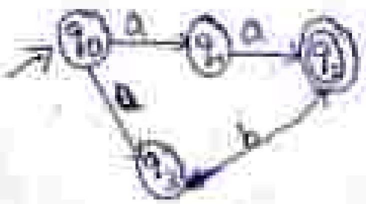
$\Sigma = \{a, b\}$

Each string starts with a and followed by 'a or b' and ending with 'a or any no. of b's.

$\underline{a}(\underline{a+b})(a+b)^*(\underline{a+b^*})$

$L(M) = \{aaa, aba, aa, ab, aaa, aaba, abaa, abab, \ldots\}$

minimum string = aa or ab



δ:

states	input a	b
$\rightarrow q_0$	$\{q_1, q_2\}$	—
q_1	q_2	—
(q_2)	—	q_3
q_3	—	q_2

3) Design an NFA that accepts set of all strings ending in 00 over $\Sigma = \{0, 1\}$

let M be a NFA like

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

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

Each string ends with 00

$(0+1)^*00$

$L(M) = \{00, 000, 100, 0000, 1100, \ldots\}$

minimum string = 00

δ:

states	input 0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	q_2	—
(q_2)	—	—

4) Design an NFA that accepts set of all strings ending in aba over $\Sigma = \{a, b\}$.

$\Rightarrow$ Let M be a NFA like

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

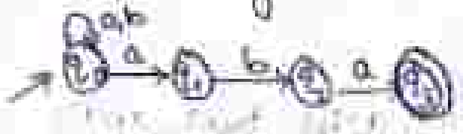
$\Sigma = \{a, b\}$

Each string ends with aba

$(a+b)^*aba$

$L(M) = \{aba, aaba, baba, aaaba, bbaba, \ldots\}$

minimum string = aba



δ:

states	input a	input b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	—	q_2
q_2	q_3	—
(q_3)	—	—

5) Given an NFA which accepts all strings with ab over $\Sigma = \{a, b\}$

Let M be NFA like

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

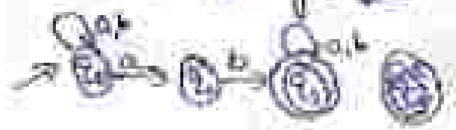
$\Sigma = \{a, b\}$

Each string have accepts all strings with ab

$(a+b)^* ab (a+b)^*$

$L(M) = \{ab, aaba, babb, \ldots\}$

minimum string = ab



δ:

states	input a	input b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	—	q_2
(q_2)	q_2	q_2

6) construct an NFA that accepts all strings over $\Sigma = \{0,1\}$ start with 0 or 1 and ends 01 or 10

$\Rightarrow$ Let M be a NFA like

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

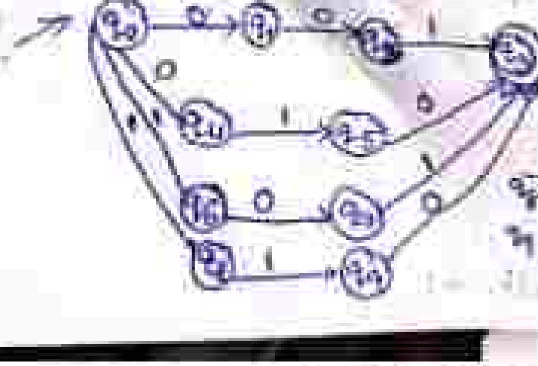
$\Sigma = \{0, 1\}$

Each string starts with 0 or 1 and ends with 01 or 10

$(0+1)(0+1)^*(01+10)$

$L(M) = \{001, 010, 101, 110, 0001, 0010, 0101, 0110, \ldots\}$

minimum string = 001 or 010 or 101 or 110



states	input 0	input 1
$\rightarrow q_0$	$\{q_1, q_4\}$	$\{q_6, q_8\}$
q_1	q_2	—
q_2	—	q_3
(q_3)	—	—
q_4	—	q_5
q_5	q_3	—
q_6	q_7	—
q_7	q_3	—
q_8	—	q_9
q_9	q_3	—

7) Construct a transition system which can accept string over the alphabets a,b,c,...z containing either cat (or) Rat

⇒ Let M be a NFA like

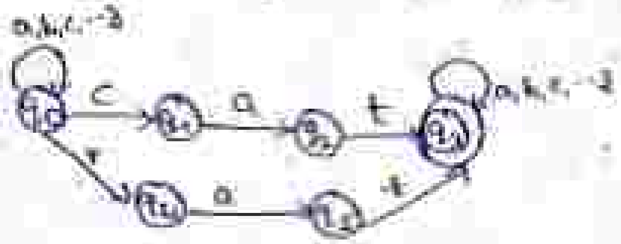
$M = (Q, \Sigma, \delta, q_0, F)$

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

each strings containing either cat (or) RAT

$(a+b+c+\cdots+z)^*(cat+rat)(a+b+c+\cdots+z)^*$

minimum string = cat or rat



δ:

states	a	b	c	...	r	...	t	...	z
→ q_0	q_0	q_0	q_0,q_1	...	q_0,q_4	...	q_0	...	q_0
q_1	q_2	-	-	-	-	-	-	-	-
q_2	-	-	-	-	-	-	q_3	-	-
q_3	q_3	q_3	q_3	-	q_3	-	q_3	-	q_3
q_4	q_5	-	-	-	-	-	-	-	-
q_5	-	-	-	-	-	-	q_3	-	-

Conversion of NFA to DFA:-

Algorithm:-

Let D be a DFA

Let N be a NFA. This algorithm is called powerset (or) subset construction Algorithm. Because for NFA (N) with 'n' states then the corresponding DFA (D) can have 2^n states.

Subset construction Algorithm:-

step 1:- Construct the start state q_0, consisting of q_0 and all the states of NFA that can be reached from q_0 by one or more transitions. Mark q_0 as unfinished.

2:- while there are unfinished states:

* Take an unfinished state S
* for each $a \in \Sigma$, $\delta(S, a) = t$ is either finished (or) unfinished state.
* mark 'S' as finished.

3:- Mark all states that contain a final state from N as final states of D.

ex:- Construct DFA from the Given NFA.

⇒ 

Sol:- The given NFA N is like

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

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

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

δ:

state	input a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	—	q_2
q_2	q_2	q_2

$q_0 = q_0$

$F = \{q_2\}$

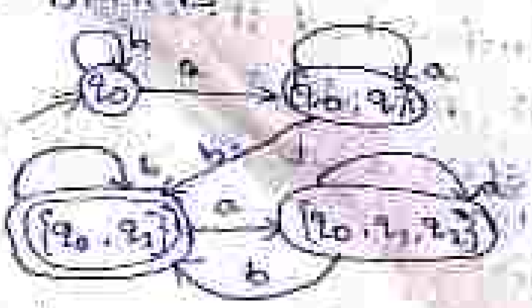
Let us consider the NFA 'D' is like

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

Apply subset construction algorithm

state	input a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$

∴ The DFA is



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

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

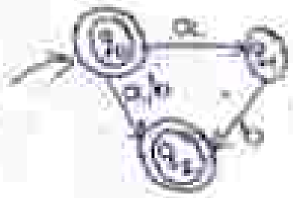
$\delta' =$

states	inputs a	b
$\rightarrow q_0$	$\{q_0,q_1\}$	q_0
$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
$\{q_0,q_2\}$	$\{q_0,q_1,q_2\}$	$\{q_0,q_1\}$
$\{q_0,q_1,q_2\}$	$\{q_0,q_1,q_2\}$	$\{q_0,q_2\}$

$q_0 = q_0$

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

Construct a DFA from the given NFA



⇒ The given NFA 'N' is like

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

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

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

δ:

states	inputs a	b
$\rightarrow q_0$	$\{q_1,q_2\}$	q_2
q_1	ϕ	q_2
q_2	ϕ	ϕ

$q_0 = q_0$

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

Let us consider the DFA 'D' is like

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

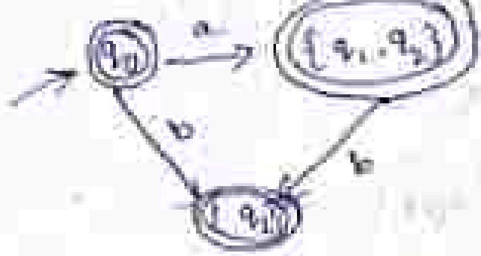
Apply subset construction algorithm

states	input a	b
$\rightarrow q_0$	$\{q_1,q_2\}$	q_2
$\{q_1,q_2\}$	ϕ	q_2
q_2	ϕ	ϕ

$q_0 = q_0$

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

∴ The DFA is



$Q' = \{q_0, [q_1, q_2], \{q_2\}\}$

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

$q_0 = q_0$

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

δ':

state	inputs a	b
q_0	$\{q_1, q_2\}$	q_2
$\{q_1, q_2\}$	ϕ	q_2
q_2	ϕ	ϕ

NFA with E-moves:

* Introduction * Representation * Elements
* External transition function * conversion of E-NFA to NFA
* E-closure

Introduction:-

* A finite state machine which contains E-moves is called E-NFA
* E-NFA is always NFA but not DFA

Definition:-

* E-NFA is a FSM where for each pair of current state and input symbol along with Epsilon have more than one next state

Elements:-

1) states 2) inputsymbols with E 3) transitions
4) initialstate 5) final state

Representation:-

E-NFA is a five tuple like $M = (Q, \Sigma, \delta, q_0, F)$,

where ~~Q = is fini~~

Q = is finite and non-empty set of states

Σ = finite and non-empty set of symbols

δ = is a transition function which is defined as $\delta: Q \times \Sigma^* \to 2^Q$

$\delta: Q \times \Sigma \cup \{E\} \to 2^Q$

Ex:-

1) consider

ε-closure (q_0) = {q_0, q_1, q_2}

ε-closure (q_1) = {q_1, q_2}

ε-closure (q_2) = {q_2}

Exr(um) Conversion of NFA with ε-moves to NFA without ε-mo and equivalence :-

1) converting NFA with ε-moves to NFA without ε-moves:

In this method we remove the ε-transitions from the given NFA and obtained NFA without ε-moves

Algorithm:-

1) find out all ε-transitions from each state in Q that will be called as ε-closure of q_i, where $q_i \in Q$.

2) δ' transitions can be obtained.

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

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

3) Repeat step 2 for each input symbol and each state given NFA

4) Finally the resultant states of NFA without ε-moves is obtained

Ex:- Convert the given ε-NFA to NFA.

sol:- Let M be a given ε-NFA like

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

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

$\Sigma = \{0, 1, 2, \epsilon\}$

δ - is a transition function

states	input			
	0	1	2	ε
→q_0	q_0	ϕ	ϕ	q_1
q_1	ϕ	q_1	ϕ	q_2
q_2	ϕ	ϕ	q_2	ϕ

$q_0 = q_0$

$F = \{q_2\}$

Now find out ε-closure for all states in the given ε-NFA.

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

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

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

Compute δ'-transitions :

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

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

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

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

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

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

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

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

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

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

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

$= \{q_1, q_2\}$

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

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

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

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

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

$= \{q_2\}$

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

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

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

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

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

$= \phi$

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

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

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

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

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

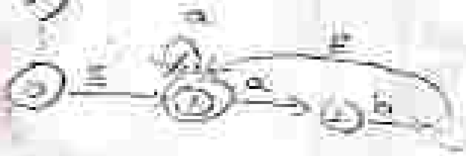
$= \{q_1,q_2\}$

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

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

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

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



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

$= \{q_2\}$

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

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

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

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

$= \phi$

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

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

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

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

$= \phi$

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

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

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

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

$= \{q_2\}$

Now the NFA without ϵ-moves M' is like

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

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

$\Sigma = \{0, 1, 2\}$

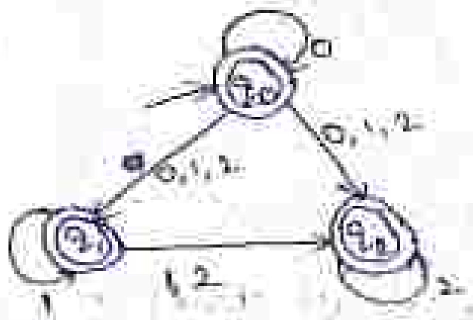
δ':

	0	1	2
$\rightarrow q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
q_1	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	$\{q_2\}$

$q_0 = q_0$

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

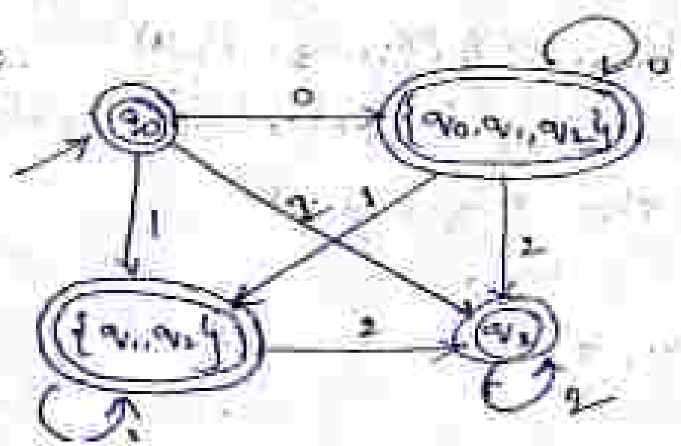
$\therefore$ NFA without ϵ-moves is



Converting NFA to DFA :-

	0	1	2
$\rightarrow q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_1, q_2\}$	ϕ	$\{q_1, q_2\}$	$\{q_2\}$
q_2	ϕ	ϕ	q_2

$\therefore$ The DFA is



2) Construct DFA from the given NFA with ε-moves

→ Let M be a given NFA like

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

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

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

δ: is a transition function

δ':

state	inputs a	b	ε
→ q_0	q_1	ϕ	$\{q_1, q_2\}$
q_1	ϕ	q_2	q_2
q_2	ϕ	ϕ	ϕ

$q_0 = q_0$

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

we can find out the ε-closure of all states in the given ε-NFA.

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

ε-closure of $(q_1) = \{q_1, q_2\}$

ε-closure of $(q_2) = \{q_2\}$

compute δ'-transitions :-

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

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

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

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

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

$= \{q_1, q_2\}$

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

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

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

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

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

$= \{q_2\}$

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

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

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

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

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

$= \phi$

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

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

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

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

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

$= \{q_2\}$

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

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

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

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

$= \phi$

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

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

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

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

$= \phi$.

Now the NFA without ϵ-moves M' is like

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

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

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

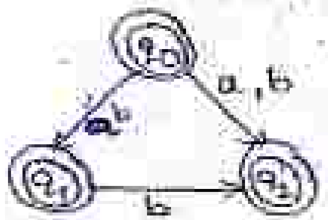
δ':

state	inputs a	b
$\rightarrow q_0$	$\{q_1, q_2\}$	q_2
q_1	ϕ	q_2
q_2	ϕ	ϕ

$q_0 = q_0$

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

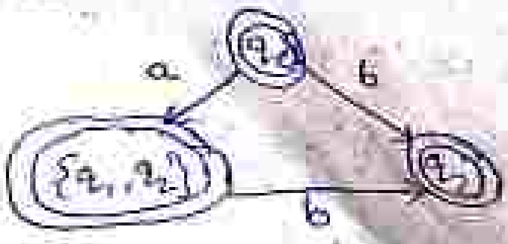
NFA without ϵ-moves is



Converting NFA to DFA

states	Inputs a	b
$\rightarrow q_0$	$\{q_1, q_2\}$	q_2
$\{q_1, q_2\}$	ϕ	q_2
q_2	ϕ	ϕ

The DFA is



Equivalence of finite state machines :-

Two finite automata's are equivalent if they accept the same set of strings over 'Σ'. Otherwise they are not equivalent.

Algorithm :-

1) We can check the equivalence of two finite state machine by using comparison table.

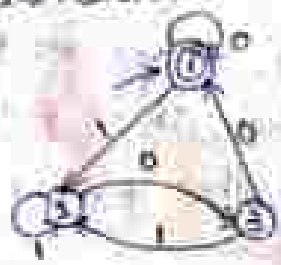
Comparison table :-

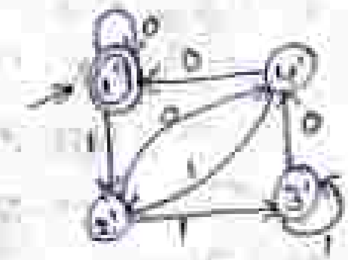
* It is similar to transition table.
* It contains (n+1) columns.
* Here, first column represents states, n columns represents input symbols.
* Entries are pair of states. Here pair of states both are final states (or) both are non-final states.

Construction of Comparison table :-

* Comparison table is starts with initial state of M and M' in the first column and consider the entry as (q_0, q_0') where, $\delta(q_0, a) = q_0$ and $\delta(q_0', a) = q_0'$.
* Repeat step 2 for each input symbol on every pair of states of M and M' until no new pairs appeared.
* If you get any pair (q_0, q_0') such that q_0 is a final state and q_0' is non-final state (or) vice versa then we terminate the process and conclude that both M and M' are not equivalent.

Ex:- check the Equivalence of the following finite automata's.





Comparison table:—

states	Input symbols 0	1
(1,1')	(1,1')	(2,2')
(2,2')	(3,4')	(2,3')
(3,4')	(1,1')	(2,2')
(2,3')	(3,4')	(2,3')

∴ The given Finite automatas are Equivalence.

Minimization (or) optimization of FA:—

The process of reducing the no. of states from given FA is called minimization (or) optimization of FA.

Equivalence states:—

The two states q_1 and q_2 are equivalent if both $\delta(q_1,a)$ and $\delta(q_2,a)$ are Final states (or) non-final states. While minimizing finite automata we first find out which two states are Equivalent then we can represent those two states by one representative state.

Minimization Algorithm:—

* we will create a set $\pi_0 = \{\{Q_1^0\}, \{Q_2^0\}\}$ where $\{Q_1^0\}$ is the set of final states and $\{Q_2^0\}$ is the set of non-final states

π_0 is 0' equivalence class.

* Now we will construct π_{K+1} from π_K.

Let Q_i^K be any subset in π_K.

If q_1 and q_2 are two states in Q_i^K.

Find out whether $\delta(q_1,a)$ and $\delta(q_2,a)$ are residing in the same Equivalence class in π_K. Then it is said that q_1,q_2 are K+1 Equivalent then Q_i^K is further divided into K+1 Equivalence classes.

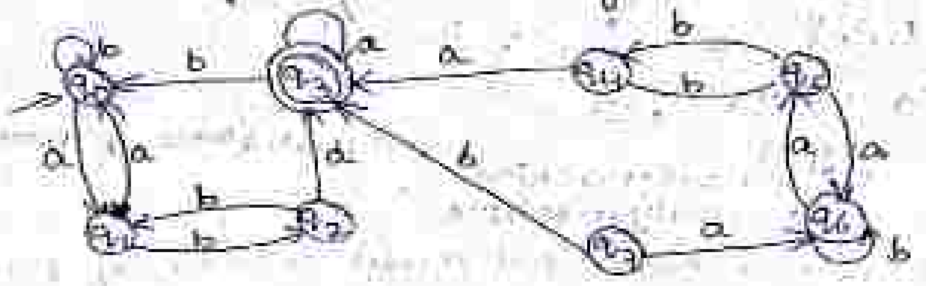
* Repeat step 2 for every Q_i^K in π_K and obtain all elements in π_{K+1}.

* Continue the above process until $\pi_n = \pi_{n+1}$ where $n = \{1,2,3,\dots\}$

* Then replace all the equivalence states in one equivalence class find representing state

This helps minimizing the given finite automata.

Ex:- Construct the state minimizing the finite automata for the following transition diagram



Sol:- The given FA, M is like

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

where

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

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

δ: is a transition function

states	Input symbols a	b
$\to q_0$	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_3	q_0
q_4	q_3	q_5
q_5	q_6	q_4
q_6	q_5	q_6
q_7	q_6	q_3

rough

$q_0 \xrightarrow{a} q_1$
$q_1 \xrightarrow{a} q_0$

$q_0 \xrightarrow{b} q_0$
$q_1 \xrightarrow{b} q_2$

$q_0 \xrightarrow{a} q_1$
$q_2 \xrightarrow{a} q_3$ ×

$q_0 \xrightarrow{a} q_1$
$q_4 \xrightarrow{a} q_3$ ×

$\pi_0 = \{\{q_3\}, \{q_0, q_1, q_2, q_4, q_5, q_6, q_7\}\}$

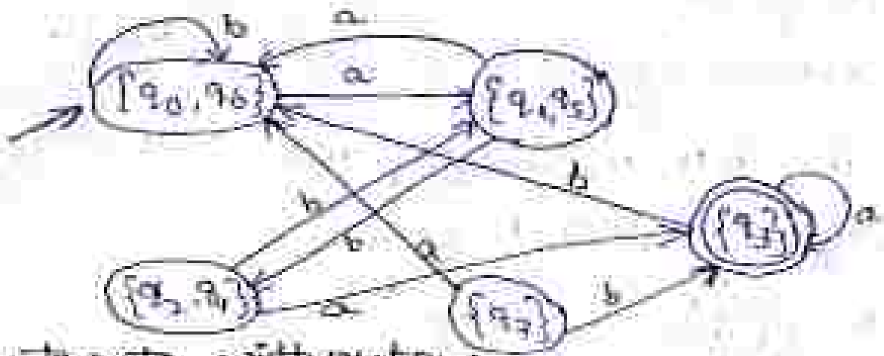
$\pi_1 = \{\{q_3\}, \{q_0, q_1, q_5, q_6\}, \{q_2, q_4, q_7\}\}$

$\pi_2 = \{\{q_3\}, \{q_0, q_6\} \{q_1, q_5\}, \{q_2, q_4\} \{q_7\}\}$

$\pi_3 = \{\{q_3\}, \{q_0, q_6\}, \{q_1, q_5\}, \{q_2, q_4\} \{q_7\}\}$

$\therefore \Pi_3 = \Pi_2$

$\therefore$ The minimized FA



Finite automata with output :-

* Introduction * types 1. moore machine 2. mealy machine * equivalence of moore and mealy machine

* Introduction :-

we know FA is a mathematical model defined by 5-tuples like $M = (Q, \Sigma, \delta, q_0, F)$.

without information about the output

after reading a string if finite automata reaches to the final state then the string is accepted by FA. Else the string is registered.

FA without output is called language acceptors

Transeducers:-

The FA with output is called Transleducers.

* The FA with output contains finalstates.
* It doesnot contains for language Acceptor.
* It cannot be used the information about output
* It provides the information about output
* It can be used for type conversion of language
* Transeducers are two types.

i) moore machine

ii) mealy machine.

i) moore machine:-

moore machine is a FA that contains set of states in which

" the output is always depends on present state only".

present state → output.

mathematically moore machine is 6 tuples like

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

→ where Q = finite and non-empty set of states.

Σ = finite and non-empty set of Input symbol

δ = It is a transition function defined as

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

Δ = It is a finite and non-empty set of output symbols (or) output Alphabets

λ = It is a output function which is defined as

$$\lambda: Q \rightarrow \Delta$$

q_0 = It is a initial state and must be $q_0 \in Q$.

Representation of moore machine:-

i) Transition diagram ii) Transition table

Transition diagram:-

$$\rightarrow (q_0/op) \xrightarrow{i/p} (q_1/op) \xrightarrow{i/p} (q_2/op)$$

Transition table:-

present states	Input symbols				output
	a	b	c	d	

Ex:- Design a moore machine for residue (3) for the input string is created as a binary number

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

residue (3) means if any number is divisible by 3 then we get the possible remainders are 0, 1, 2

$\therefore \Delta = \{0,1,2\}$

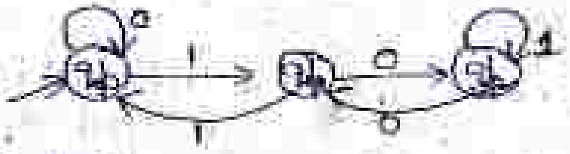
The possible state are $\{q_0, q_1, q_2\}$

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

$\therefore \delta$: transition function is defined as

δ	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

∴ the moore machine is



Transition table :-

present state	I/p symbols 0	1	output
→ q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

Note :- In moore machine (n+1) output symbols can be produc... as 'n' input symbols.

Mealy machine:-

It is a finite automata contains set of states in which "the output is always depends on present state and i/p symbol"

mathematically mealy machine defined by 6 tuples like $m = (Q, \Sigma, \delta, \Delta, \lambda, q_0)$

where,

$Q \rightarrow$ finite and non empty set of states

$\Sigma \rightarrow$ finite and non empty set of states and input symbols.

δ is a transition function is defined as

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

Δ is a finite and non empty set of states and output symbols.

λ is a output function is defined as,

q_0 is the initial state must be $q_0 \in Q$

Representation :- In two ways

1. Transition diagram 2. Transition Table

① Transition diagram

1. Transition Table

Present state	Input symbols$_1$		Input symbols$_2$	
	Next state	output	Next state	output

Design a mealy machine for residue mode 3 in which the input is treated as a binary machine.

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

Residue mode 3 means if any number is divisible by 3 then the possible remainders are 0,1,2

$\Delta = \{0,1,2\}$

The possible states are $\{q_0, q_1, q_2\}$

$\therefore \delta$ is transition function

δ: state	Input symbols 0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

The mealy machine is

Transition table

present state	input symbol$_1$ (0) Next state	output	Input symbol$_2$ (1) Next state	output
$\rightarrow q_0$	q_0	0	q_1	1
q_1	q_2	2	q_0	0
q_2	q_1	1	q_2	2

Note:- In mealy machine 'n' output symbols can be produced for 'n' input symbols.

Equivalence of mealy and moore machine:-

conversion of moore to mealy machine

conversion of mealy to moore machine

conversion of moore and mealy machine:-

convert the following moore machine to mealy machine

The given moore machine is M like

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

$Q = \{q_0, q_1\}$

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

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

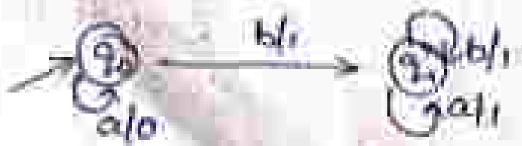
λ : output function

present state	Input symbols a	b	output
$\rightarrow q_0$	q_0	q_1	0
q_1	q_1	q_1	1

Now the transition table for mealy machine is

present state	Input symbol₁ (a)		Input symbol₂ (b)	
	next state	output	next state	output
$\rightarrow q_0$	q_0	0	q_1	1
q_1	q_1	1	q_1	1

Transition diagram for mealy machine is



conversion of mealy to moore machine:-

Convert the following mealy machine to moore machine.

The given mealy machine is m like

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

$Q = \{q_0, q_1\}$

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

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

δ:

present state	a		b	
	N.S	o/p	N.S	o/p
$\to q_0$	q_0	0	q_1	1
q_1	q_1	1	q_1	1

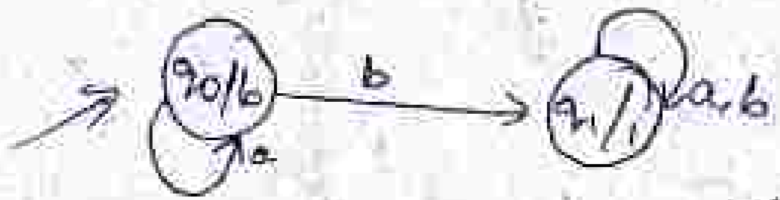
δ : e

	a	b
q_0	q_0	q_1
q_1	q_1	q_1

Now moore machine table.

Present state	Input symbol a	Input symbol b	output
→ q_0	q_0	q_1	0
q_1	q_1	q_1	1

Transition diagram for moore machine is



convert the following mealy machine to moore machine

P.S	a		b	
	N.S	o/p	N.S	o/p
$\rightarrow q_0$	q_0	0	q_1	1
q_1	q_0	1	q_1	0

$q_0 \begin{cases} 0 \rightarrow q_{00} \\ 1 \rightarrow q_{01} \end{cases}$ $q_1 \begin{cases} 0 \rightarrow q_{10} \\ 1 \rightarrow q_{11} \end{cases}$

Transition table for moore machine

P.S	a	b	output
$\rightarrow q_{00}$	q_{00}	q_{11}	0
$\rightarrow q_{01}$	q_{00}	q_{11}	1
q_{10}	q_{01}	q_{10}	0
q_{11}	q_{01}	q_{10}	1

Applications and limitations of FA:-

limitations:-

* FA contains and i/p buffer with limited (or) finite no. of locations. i.e; it has a limited amount of memory for storing i/p data.
* It recognize a finite length of i/p string.
* It can moves its read/write head in either left to right (or) right to left direction only.

Applications:-

* FA is used to design lexical analyzer.
* FA is used to create text editors.
* FA is used to spellchecking.
* FA is used to design sequential circuits.