UNIT-1

FORMAL LANGUAGE AND REGULAR EXPRESSIONS

Basics:-

Symbol:- Symbol is an abstract entity that we shall not define formally, just as "point" and "line" are not defined in geometry.

letters and digits are examples of frequently used symbols.

String:- is a finite sequence of symbols chosen from Σ

Ex:- a, b and c are symbols and abcd is String.

The length of string w, denoted by $|w|$, is the number of symbols composing the string.

$\therefore$ $w = abcd$ $\therefore |w| = 4$

$\therefore$ The empty string, denoted by ϵ, is the string consisting of zero symbols. $\therefore |\epsilon| = 0$

- A **Prefix** of a string is any number of leading symbols of that string.

  Ex:- abc has Prefixes ϵ, a, ab, and abc.

- **Suffix:-** is any number of trailing symbols

  Ex:- abc has suffixes are ϵ, c, bc and abc.

Operations on languages

Concatenation:- of two strings is formed by writing the first followed by the second, with no intervening space.

$\therefore$ if w and x are two string, then wx is concatenation of two strings.

if L_1, L_2 are two languages then we can generate a new language L_1L_2 which is defined as

$L_1L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$.

• operator.

• i.e $\epsilon w = w\epsilon = w$ for each string w.

• An alphabet is a finite* set of symbols which are used to form words in language. denote as Σ (sigma)

Ex:- Alphabet might be set like $\{a, b\}$

Here we denote alphabet using the symbol Σ.

$\therefore \Sigma = \{a, b\}$ so then string over Σ can be a, ab, bbaa..

• For some alphabet Σ, we use Σ^* to denote the set of all possible strings over Σ

Applying * to some set is called the closure operation.

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

$\Sigma^* = \{a, b\}^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa \cdots\}$

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots \qquad \Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots$

• A language over Σ will be a set of strings over Σ.

So it will be some subset of Σ^*.

So Examples of language might be.

$\{\epsilon, a, aa, bb\}$

$\{x \in \{a, b\}^* \mid |x| \leq 3\}$

$\{x \in \{a, b\}^* \mid |x| = 4\}$

$\{x \in \{a, b\}^* \mid x$ has equal number of a's and b's$\}$

$\{x \in \{a, b\}^* \mid x$ always ends with an a$\}$

Language- A set of strings is called a language. It is also called formal language. denoted by L.

$L = \{01\}$ $L = \{a, b\}$

A Language can be finite (or) infinite.

A Language can be $L \subseteq \Sigma^*$ if

- A set is a group of objects represented as a unit. Set's may contain any type of object, including members, figures, and even other sets.
The objects in a set are called its elements (or) members.

∴ Set {1, 2, 3, 17}

∴ A is subset of B $A \subseteq B$ means A if every member of A also is a member of B.

- The set with "0" members is called the empty set and is written ϕ.
- When we want to describe a set containing element's according to some rule
we write {n | rule about n}
Thus {n | n = m² for some m ∈ N} means set of perfect squares.

- Functions :- function is an object that set's up an i/p and o/p relationship. A function takes an input and produces an o/p.
∴ f(a) = b

- The set of possible inputs to the function is called it's domain.
- The outputs of a function come from a set is called it's range.

$$f : D \rightarrow R$$

Language :- Collection of strings

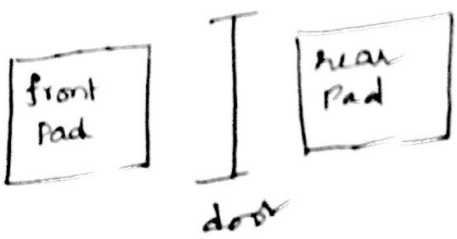
REGULAR LANGUAGES

Automata theory is study of abstract machines (abstract Mathematical m/c) and the computational problems that can be solved using these machines. These abstract machines called automata.

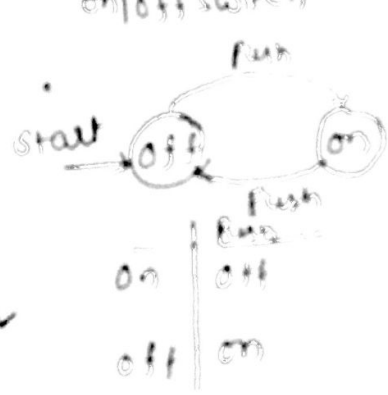
FINITE AUTOMATA :- Finite automata are good models for computers with an extreamely limited amount of memory. What can a computer do with such a small memory? In fact, we interact with such computers all the time, as they lie at the heart of various electro mechanical devices.

Ex:- Controller for an automatic door

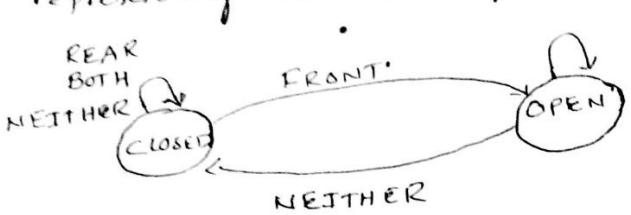
* often found at a supermarket entrances and exits automatic doors swing open when sensing that a person is approching.
* Automatic door has a pad in front to detect the presence of a person about to walk through the doorway.
* Another pad is rear of the doorway so that the controller can hold the door open long enough for the person to pass all the way through and also so that the door does not strike some one standing behind it as it opens.



Top view of automatic door



• The controller is in either of two states: "OPEN" or "CLOSED" representing corresponding condition of door.



→ State diagram for automatic door controller

As shown above figures, there are four possible i/p conditions.

"FRONT": means that a person is standing on the pad in front of the doorway.

"REAR": Person is standing on the Pad ~~in front~~ Rear of the doorway

"BOTH": People are standing on both pads

"NEITHER": None is standing on either pad.

State table:-

	NEITHER	FRONT	REAR	BOTH
CLOSED	CLOSED	OPEN	CLOSED	CLOSED
OPEN	CLOSED	OPEN	OPEN	OPEN

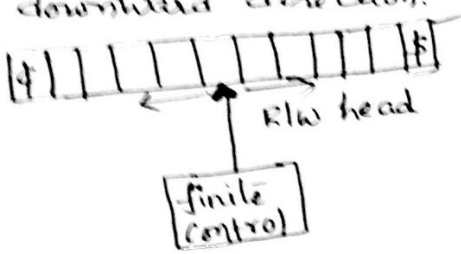
State transition table for automatic door controller.

• This controller is a computer that has just a single bit of memory, capable of recording which of the two states the controller is in.

• We will now take a close look at finite automa from Mathematical perspective.

Block diagram of Finite Automata

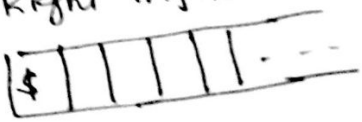
- The block diagram of FA consists of a tape (O/P tape which is divided into no of cells.
- Each cell is a square used to store a single symbol from the I/P alphabet set. It consists of read/write head used to read a symbol from the tape (or) write a symbol to the tape.
- Another component is finite control. It is control or program which transfers the read/write head either upward or downward direction.



- If tape consists of end markers, it is said to be a finite tape, otherwise it is said to be an infinite tape.

Ex Left infinite

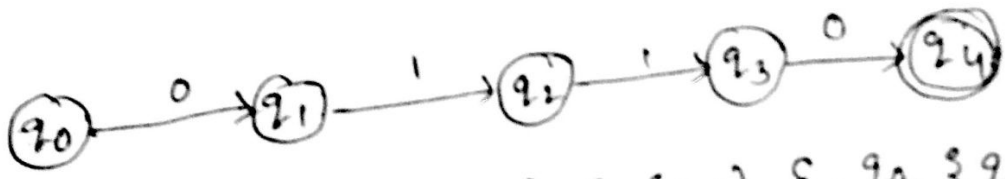
Right infinite



EX ① Construct a Finite automata accepting the string

0110

Sol:-



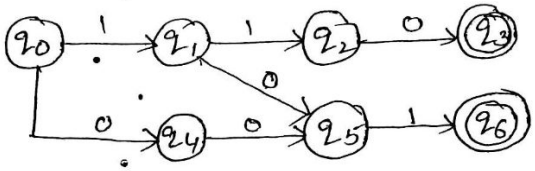
$F.A = (Q = \{q_0, q_1, q_2, q_3, q_4\}, \Sigma = \{0,1\}, \delta, q_0, \{q_4\})$

where $\delta = \delta(q_0, 0) = q_1 \quad \delta(q_2, 1) = q_3$

$\delta(q_1, 1) = q_2 \quad \delta(q_3, 0) = q_4$

Ex:② Construct a F.A accepting the strings 110, 001, 101 these 3 strings

Sol:-



F.A = (Q = {q0, q1, q2, q3 ---- q6}, Σ = {0,1}, δ, q0, {q3, q6})

Where δ is defined as

δ(q0,1) = q1 δ(q0,0) = q4

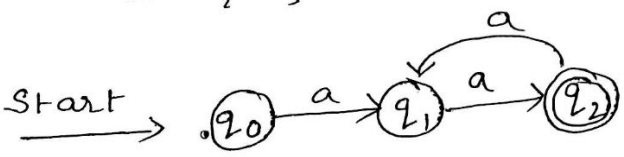
δ(q1,1) = q2 δ(q4,0) = q5

δ(q2,0) = q3 δ(q5,1) = q6

Ex:③ Construct a F.A which accepts the following language L = { an/n = 2,4,6,8 ---- }

Sol:- Hear L = { aa, aaaa, aaaaaa ---- }

Σ = { a }



F.A = (Q = { q0, q1, q2 }, Σ = { a }, δ, q0, { q2 })

Where δ is defined as δ(q0,a) = q1

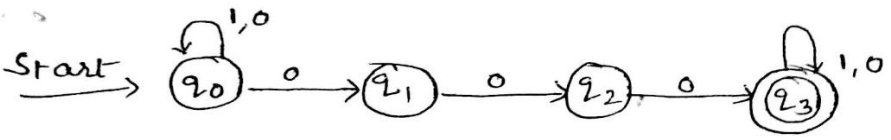
δ(q1,a) = q2

δ(q2,a) = q1

Ex 4:- Construct FA for the language L = { x/x consist's 3 consecutive 0's as a substring over Σ = {0,1} }

Sol:- Hear L = { 000, 1000, 0001, 11000, 10001 ---- }

Σ = {0,1}



F.A = $(\{q_0, q_1, q_2, q_3\}, \Sigma = \{0, 1\}, \delta, q_0, \{q_3\})$

where δ is defined as

$\delta(q_0, 0) = \{q_0, q_1\}$ $\quad \delta(q_1, 0) = q_2$ $\quad \delta(q_3, 0) = q_3$

$\delta(q_0, 1) = q_0$ $\quad \delta(q_2, 0) = q_3$ $\quad \delta(q_3, 1) = q_3$

KLEEN STAR (OR) KLEEN CLOSURE :-

The Kleen star of a set S is the ~~element~~ concatenation of the elements of the set S any number of times. It is denoted by S^*. By default ϵ is an element of S^*.

Ex:- If $S = \{a\}$ then $S^* = \{\epsilon, a, aa, aaa \cdots\}$

If $S_1 = \{a, b\}$ then $S_1^* = \{\epsilon, a, b, aa, bb, ab, ba \cdots\}$

If $S_2 = \{a, b, c\}$ then $S_2^* = \{\epsilon, a, b, c, aa, bb, cc, abc \cdots\}$

POSITIVE CLOSURE (OR) ABSOLUTE CLOSURE :-

Positive closure of a set S is the concatenation of the elements of S any number of times excluding ϵ. It is denoted by S^+

$\therefore$ i.e $\boxed{S^+ = S^* - \{\epsilon\}}$

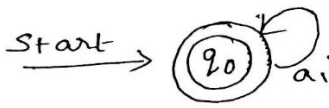
Ex:- If $S = \{a\}$ then $S^+ = \{a, aa, aaa\}$

The

Ex:-1 Construct a F.A for L^* and L^+ where $L = \{a, b\}$.

Sol:- Given $L = \{a, b\}$

(i) $L^* = \{\epsilon, a, b, aa, bb, ab \text{ ----}\}$



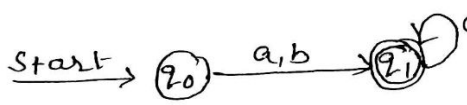
F.A for L^* is

$F.A = (Q = \{q_0\}, \Sigma = \{a, b\}, \delta, q_0, \{q_0\})$

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

$\delta(q_0, b) = q_0$

(ii) $L^+ = \{a, b, aa, bb, ab \text{ ----}\}$



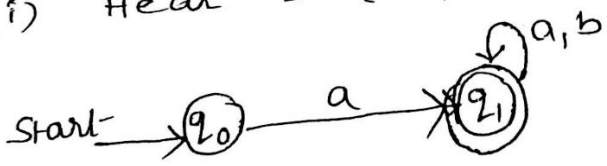
$F.A = (Q = \{q_0\}, \Sigma = \{a, b\}, \delta, q_0, \{q_1\})$

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

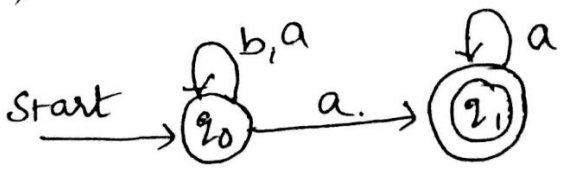
$\delta(q_1, a) = q_1$ $\delta(q_1, b) = q_1$

Ex: 2: construct F.A which accepts all the strings over $\Sigma = \{a, b\}$ where every string (i) begining with 'a' (ii) Ending with 'a'.

Sol: (i) Hear $L = \{a, aa, ab, abb, aba \text{ ----}\}$



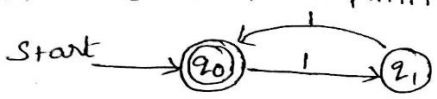
(ii) Hear $L = \{a, aa, ba, bba, aba \text{ ----}\}$



Ex:-③ Construct a F.A which accepts the language whose strings consist (i) Even number of 1's

(ii) odd no of 1's

Sol:-(i) $L = \{ \in, 11, 1111, 111111 \cdots \}$



(ii) $L = \{ 1, 111, 11111 \cdots \}$

DETERMINISTIC FINITE AUTOMATA :-

- DFA Consist's of 5 tuples $(Q, \Sigma, \delta, q_0, F)$

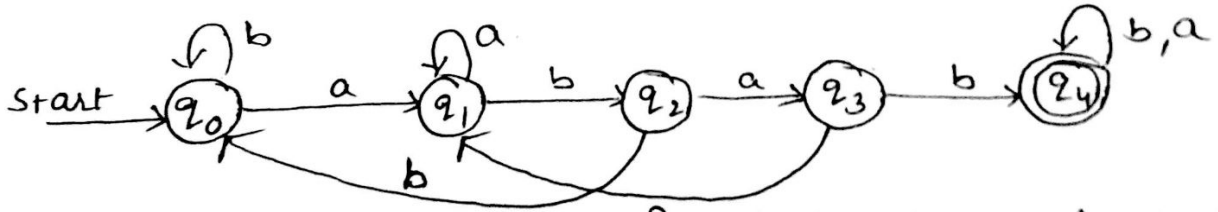
Q – finite set of states
Σ – finte i/p alphabet set
q_0 – is the starting state
F – final state $\therefore F \subseteq Q$
$\delta : Q \times \Sigma \rightarrow Q$ has a unique transition each and every state takes all the I/p Symbols from Σ only once.

Ex:- 

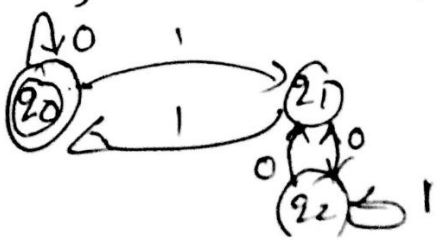
is a DFA

Example :-① Construct a DFA which accepts "abab" as a substring over Σ when $\Sigma = \{a, b\}$.

Sol:- Hear $L = \{$ abab, aabab, ababb, aababb $\cdots \}$
abbabab, ababbabab $\cdots \}$



Construct DFA for a language $L = \{ x \mid x \in \Sigma, x \bmod 3 = 0$ when x as a binary number (or) all the strings divisible by 3 over $\Sigma = \{0,1\}$

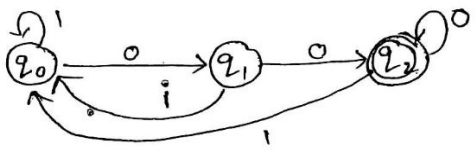


ie $L = \{ 0, 11, 110, 1001, 1100, 1111$
$10010, 10101 \cdots \}$

Ex:-2: Construct DFA consisting of set of all strings ending with 00 over $\Sigma = \{0,1\}$

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

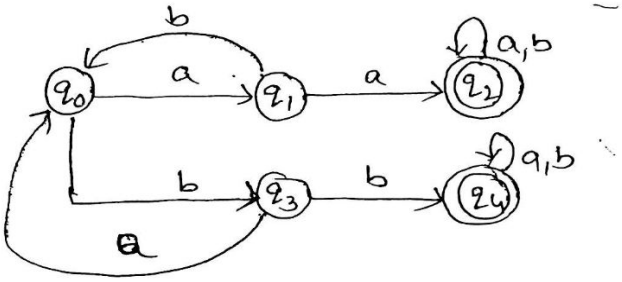
Hence $L = \{00, 100, 1100, 0000, 10100, 00100, 01000, \ldots\}$



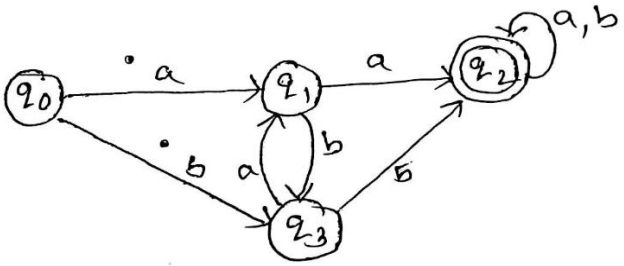
Ex:-③ Construct the DFA for the language

$L = \{ w/w$ has either aa (or) bb as a substring over $\Sigma = \{a,b\} \}$

Sol:- $L = \{ aa, bb, aaa, bbb, abb, bba, aab, baa, \ldots \}$



(OR)

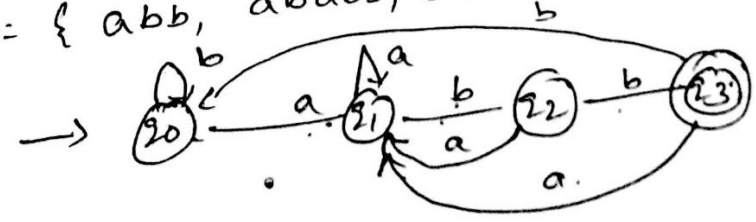


Ex4:- Construct DFA for $L = (a/b)^* abb$ over input set $\{a,b\}$

Sol:- $L = \{ abb, ababb, baabb \ldots \}$

(1) ~~abb~~

(2) ~~abba~~



$\leftarrow \{ abbabb$
$abbbabb \}$

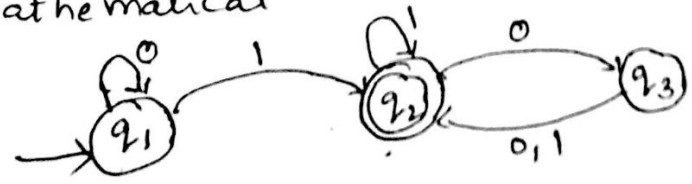
FINITE AUTOMATA:-

- consists of a finite set of states and a set of transitions from state to state that occure on i/p symbols chosen from an alphabet Σ. for each input symbol there is exactly one transistion out of each state.
- one state, usually denoted by q_0, is the initial state, in which automation starts
- finite Automation is 5-tuple. $(Q, \Sigma, \delta, q_0, F)$, where
  1. Q is a finite set called the states
  2. Σ is a finite set called the alphabet
  3. $\delta : Q \times \Sigma \to Q$ is the transistion function
  4. $q_0 \in Q$ is the start state
  5. $F \subseteq Q$ is the set of accept states (or) final states

- Hear transistion function is frequently denoted δ, to define the rules for moving.

Ex:- $\delta(q_1, 1) = q_2$ this notation is a kind of Mathematical short hand.

Ex:-



we can describe M_1 formally by writing $M_1 =$ $(Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. $\delta =$

3. δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

4. q_1 is start state

5. $F = \{q_2\}$

In our example, let

$A = \{w/w$ contains at least one 1 and even number of 0's follow the last 1.

- When this automation receives an input string as 1001 it processes the string and produces an output
- The output is either accept (or) reject.
- Evalution from left to right of string

$\therefore \delta(q_0, 1) \Rightarrow q_2$ and $\delta(q_2, 1) = q_2$

$\therefore \delta(q_0, 11) \Rightarrow \delta(\delta(q_0, 1), 1) = \delta(q_2, 1) = q_2$

$\therefore \delta(q_2, 01) \Rightarrow \delta(\delta(q_2, 0), 1) = q_2$

(OR)

$\delta(\delta(\delta(\delta(q_0,1),1),0),1)$
$\Rightarrow \delta(\delta(\delta(q_2,1),0),1)$
$\Rightarrow \delta(\delta(q_2,0),1)$
$\Rightarrow \delta(q_3,1)$
$\Rightarrow q_2$ final st.

- Accept because M_1 is an accept state q_2 at the end of input.
- So Experimenting with this M/c on a variety of input strings reveals that it accepts the string 1, 01, 11 and 0101 010101.

In fact M_1 accept the string that end with 1.

Consider transition diagram below. In our formal Notation this FA is denoted $M = (Q, \Sigma, \delta, q_0, F)$

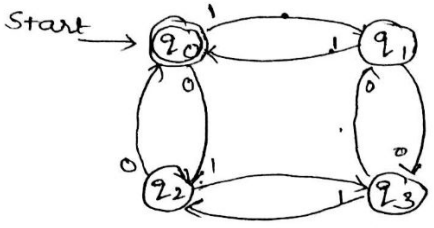


fig:

Ans:-

(1) $Q = \{q_0, q_1, q_2, q_3\}$

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

(3) $\delta : Q \times \Sigma \longrightarrow Q$

States	Inputs 0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

$\longrightarrow \delta(q,a)$ for the FA

(4) $q_0 \in Q$

(5) $F = \{q_0\}$

Suppose 110101 is i/p to M. we note that

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

$\Rightarrow$ Thus $\delta(q_0, 11) = \delta(\delta(q_0, 1), 1) = \delta(q_1, 1) = q_0$

$\Rightarrow \delta(q_0, 0) = q_2$ $\quad (\because \delta(q_0, 110) \Rightarrow \delta(\delta(q_0, 11), 0) = \delta(q_0, 0) = q_2)$

Continuing in this fashion

$\delta(q_0, 1101) = q_3$ $\quad (\because \delta(q_2, 1) = q_3)$

$\delta(q_0, 11010) = q_1$ $\quad (\because \delta(q_3, 0) = q_1)$

$\delta(q_0, 110101) = q_0$ $\quad (\because \delta(q_1, 1) = q_0)$

Thus 110101 is in $L(M)$. $L(M)$ is set of strings with even number of 0's and even number of 1's.

Ex: 

formal description $M = (\{q_1, q_2\}, \{0,1\}, \delta, q_1, \{q_2\})$, labels of transition alphabet

transition function δ is

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

NFA has or

or NF

of

∴ After trying a few more example, you would see that

M_2 accepts all strings that end in a 1.

∴ $L(M) = \{w / w \text{ ends in a } 1\}$.

Ex: 

$L(M) = \{w / w \text{ is the empty string } (\varepsilon) \text{ or end in a } 0\}$

NONDETERMINISM:

- So far in our discussion, every step of computation follows in a unique way from the preceding step.
- When the machine is in given state and reads the next input symbol, we know what the next state will be - it is determined. We call this as deterministic computation.
- In a nondeterministic machine, several choices may exist for the next state at any point.

Nondeterminism is a generalization of determinism

So every deterministic finite automation is automat a nondeterministic finite automation.

FA, labels on the transition arrows are symbols from the alphabet.

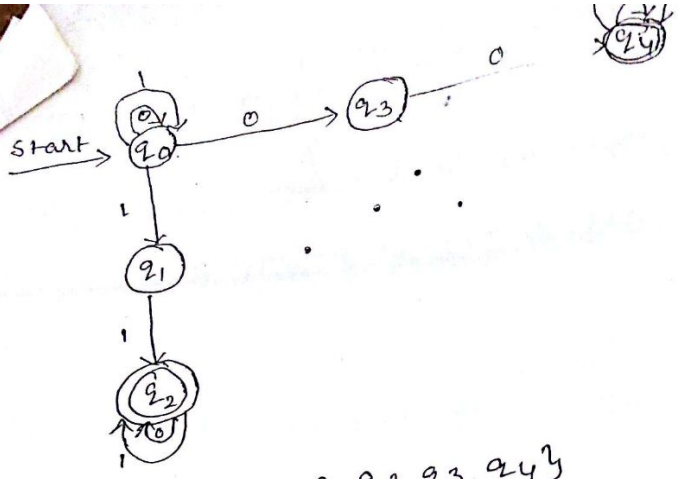
- NFA has an arrow with the label ϵ. In general, an NFA may have arrows labeled with members of the alphabet or ϵ.

FORMAL DEFINITION OF A NONDETERMINISTIC FINITE AUTOMATA

- Formal defination of ~~DFA~~ NFA is similar to DFA Both have states, an input alphabet, a transistion function a start state, a collection of accept states.
- How ever they differ in transistion function
- In an NFA transition function takes a state and an input symbol or the empty string and ~~set~~ produces the set of possible next states.

- NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where
  1. Q is finite set of states
  2. Σ is a finite alphabet
  3. $\delta: Q \times \Sigma_\epsilon \rightarrow P(Q)$ ($P(Q) = 2^Q$) is the transistion function
  4. $q_0 \in Q$ is start state
  5. $F \subseteq Q$ is the set of accept states.

$P(Q)$ is powerset of Q, the set of all subsets of Q. i.e each and every state takes max up to 2^Q transitions.



1. $Q = \{q_0, q_1, q_2, q_3, q_4\}$
2. $\Sigma = \{0, 1\}$
3. δ is given as

States	Input 0	Input 1
q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	ϕ	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	ϕ
q_4	$\{q_4\}$	$\{q_4\}$

4. q_1 is start state, and
5. $F = \{q_2, q_4\}$.

* Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w a string over the alphabet Σ. Then we say that N accepts w if we can write w as $w = y_1 y_2 \cdots y_m$ where each y_i is member of Σ and a sequence of states $r_0, r_1, \cdots r_m$ exists in Q with the following three conditions.

1. $r_0 = q_0$
2. $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, \ldots m-1$, and
3. $r_m \in F$

Condition 1 says M/c starts out in the start state.

condition 2 says that state r_{i+1} is one of the allowable next states when N is in state r_i and reading y_{i+1}

- observe that $\delta(r_i, y_{i+1})$ is the set of allowable next states and so we say that r_{i+1} is a member of that set.
- condition 3 says that it is accept state.
- $\delta(T, w) = \bigcup_{q \in p} \delta(q, w)$

→ Let we take input be 01001

$\delta(q_0, 0) = \{q_0, q_3\}$

$\delta(q_0, 01) = \delta(\delta(q_0, 0), 1) = \delta(\{q_0, q_3\}, 1)$

$= \delta(q_0, 1) \cup \delta(q_3, 1)$

$= \{q_0, q_1\}$

Similarly we compute

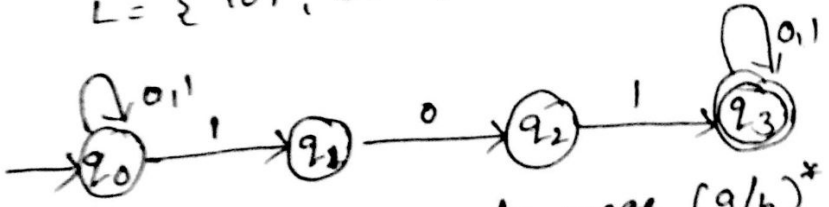
$\delta(q_0, 010) = \{q_0, q_3\}$, $\delta(q_0, 0100) = \{q_0, q_3, q_4\}$

$\therefore \delta(q_0, 01001) = \{q_0, q_1, q_4\}$

~~Block diagram of FINITE AUTOMATA~~

Ex:- Construct NFA which accepts 101 as a substring over $\Sigma = \{0, 1\}$

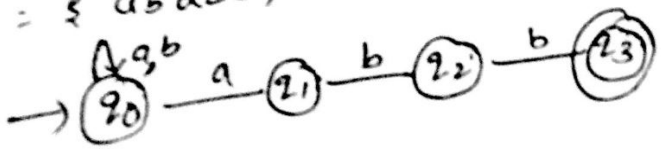
Sol:- $L = \{101, 0101, 1010, 01010, 1101, 00101 \cdots\}$



Construct NFA for the language $(a/b)^* abb$

:- $L = \{abb, aabb, baabbb, \cdots\}$

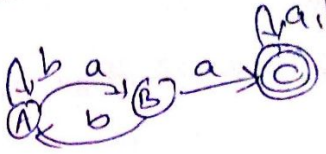
Sol:-



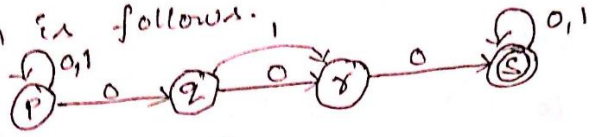
- Conversion from NFA to DFA :-

Ex:- Construct DFA for the NFA given bellow $M = (\{P, q, r, s\}, \{0, 1\}, \delta, P, \{s\})$ where δ is defined as follows.

δ	0	1
P	$\{P, q\}$	P
q	r	r
r	s	-
(s)	s	s

② 

Sol:- NFA is follows.



- Hear $\Sigma = \{0, 1\}$
- Starting state is P
- Final state $F = s$
- Let us assume that $P = a \rightarrow$ 'a' is the starting state of the DFA equivalent to the given NFA
- Let δ' be the transition function of the DFA.
- $\delta'(a, 0) = \delta(P, 0) = \{P, q\} \longrightarrow b$ (say)
- $\delta'(a, 1) = \delta(P, 1) = \{P\} \longrightarrow c$ (say)
- $\delta'(b, 0) = \delta(\{P, q\}, 0) = \delta(P, 0) \cup \delta(q, 0)$
  $= \{P, q\} \cup \{r\} = \{P, q, r\} \longrightarrow d$ (say)

$\delta'(b, 1) = \delta(\{P, q\}, 1)$
$= \delta(P, 1) \cup \delta(q, 1)$
$= \{P\} \cup \{r\}$
$= \{P, r\} \longrightarrow e$ (say)

$\delta'(c,0) = \delta(p,0) = \{p,q\} \longrightarrow b$

$\delta'(c,1) = \delta(p,1) = p \longrightarrow c$

$\delta'(d,0) = \delta(\{p,q,r\},0)$

$= \delta(p,0) \cup \delta(q,0) \cup \delta(r,0)$

$= \{p,q\} \cup \{r\} \cup \{s\}$

$\Rightarrow \{p,q,r,s\} \longrightarrow f$ (say)

$\delta'(d,1) = \delta(\{p,q,r\},1)$

$= \delta(p,1) \cup \delta(q,1) \cup \delta(r,1)$

$= \{p\} \cup \{r\} \cup \phi$

$= \{p,r\} \longrightarrow e$

$\delta'(e,0) = \delta(\{p,r\},0)$

$= \delta(p,0) \cup \delta(r,0)$

$= \{p,q\} \cup \{s\}$

$= \{p,q,s\} \longrightarrow g$ (say)

$\delta'(e,1)$

$= \delta(\{p,r\},0) \Rightarrow \delta(\{p,r\},1)$

$= \delta(p,0) \cup \delta(r,0) \Rightarrow \delta(p,1) \cup \delta(r,1)$

$= \{p,q\} \cup \{s\} \Rightarrow \{p\} \cup \phi$

$= \{p,q,s\} \Rightarrow p \longrightarrow c$

$\delta'(f,0) = \delta(\{p,q,r,s\},0)$

$\Rightarrow \delta(p,0) \cup \delta(q,0) \cup \delta(r,p) \cup \delta(s,0)$

$= \{p,q\} \cup \{r\} \cup \{s\} \cup \{s\}$

$= \{p,q,r,s\} \longrightarrow f.$

$\delta'(f,1) = \delta(\{p,q,r,s\},1)$

$= \delta\{p,1) \cup \delta(q,1) \cup \delta(r,1) \cup \delta(s,1)$

$\Rightarrow \{p\} \cup \{r\} \cup \phi \cup \{s\}$

$\Rightarrow \{p\ r,s\} \longrightarrow h$ (say)

$\delta'(q,0) = \delta(\{p,q,s\},0)$

$= \delta(p,0) \cup \delta(q,0) \cup \delta(s,0)$

$= \{p,q\} \cup \{r\} \cup \{s\}$

$= \{p,q,r,s\} \longrightarrow f$

$\delta'(q,1) = \delta(\{p,q,s\},1)$

$= \delta(p,1) \cup \delta(q,1) \cup \delta(s,1)$

$= \{p\} \cup \{r\} \cup \{s\}$

$= \{p,r,s\} \longrightarrow h$

$\delta'(h,0) = \delta(\{p,r,s\},0)$

$= \delta(p,0) \cup \delta(r,0) \cup \delta(s,0)$

$= \{p,q\} \cup \{s\} \cup \{s\}$

$= \{p,q,s\} \longrightarrow g$

$\delta'(h,1) = \delta(\{p,r,s\},1)$

$= \delta(p,1) \cup \delta(r,1) \cup \delta(s,1)$

$= \{p\} \cup \phi \cup \{s\} = \{p,s\} \longrightarrow i$

$\delta'(i,0) = \delta(\{p,s\},0)$

$= \delta(p,0) \cup \delta(s,0)$

$= \{p,q\} \cup \{s\}$

$= \{p,q,s\} \longrightarrow g$

$\delta'(i,1) = \delta(\{p,s\},1)$

$= \delta(p,1) \cup \delta(s,1) = \{p,s\} \longrightarrow i$

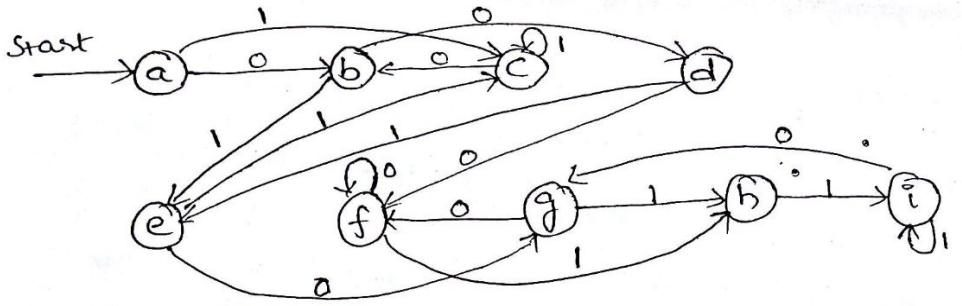
If the combination of NFA states consist of a final state of NFA, thus the corresponding DFA state is also a final state.

The final state of DFA are

$F = \{f, g, h, i\}$

DFA = (Q = {a, b, c, d, e, f, g, h, i}, Σ = {0, 1},
δ', a, F = {f, g, h, i})

where δ' is defined as ~~follows~~ above



Verification

Let us consider an arbitary string 110011

Consider δ(P, 110011)
= δ(P, 10011)
= δ(P, 0011)
= δ(P, 011)
= δ(P, 11)
= δ(P, 1)
= P The string is not accepted by the NFA (∵ P is not a final state of NFA)

Consider δ'(a, 110011) = δ'(c, 10011)
= δ'(c, 0011)
= δ'(b, 011)
= δ'(d, 11)
= δ'(e, 1)
= c.

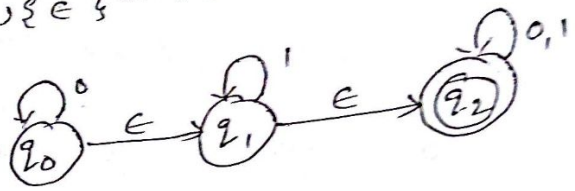
∴ The string is not accepted by DFA also
∴ (c is not the final state of DFA)
∴ Hence DFA is equivalent to given NFA

NOTE:- If the arbitary string is accepted by ... be accepted by DFA also & if it is not accepted by NFA, then it must not be accepted by DFA. Then only both NFA & DFA are said to be equivalent.

NFA - ϵ Moves:

- NFA with ϵ moves consists of 5 tuples $(Q, \Sigma, \delta, q_0, F)$ where δ is defined as

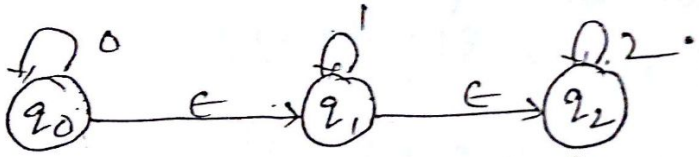
$$\delta : Q \times \Sigma \cup \{\epsilon\} \longrightarrow 2^Q$$

Ex:① 

Ex:-② Construct an NFA-ϵ Moves for $0^* 1^* 2^*$

Sol:- $\{\epsilon, 0, 00, 000, 0000\}$ $\{\epsilon, 1, 11, 111, 1111 \cdots\}$

$\{\epsilon, 2, 22, 222, 2222 \cdots\}$



	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

$\delta(q, a)$ for the NFA.

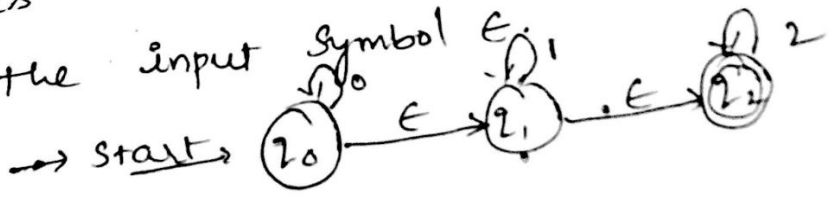
in above transition diagram such that an NFA accepting language consisting of any number of 0's followed by any number of 1's and 2's.

- As alway we say NFA accepts a string w if there is some path labeled w from initial state to final state.
- ofcourse, edges labeled ϵ may be included in the Path, although the ϵ do not appear explicitly in it.

Ex:- The word 002 is accepted by the NFA in above fig by the path $q_0, q_0, q_0, q_1, q_2, q_2$ with arc's labeled 0, 0, ϵ, ϵ, 2.

ϵ-closure :- ϵ-closure is defined on any set consisting of at least one element. Hear set means set of states.

- ϵ-closure of any state is nothing but the various states reachable from that particular state with the input symbol ϵ.



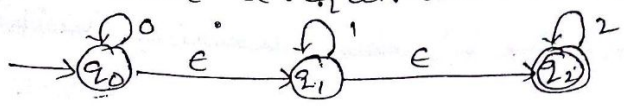
(i) ϵ-closure(q_0) = {q_0, q_1, q_2}

(ii) ϵ-closure(q_1) = {q_1, q_2}

(iii) ϵ-closure(q_2) = {q_2}

Conversion of NFA's with ϵ to NFA (equivalence of NFA-ϵ to NFA):-

Ex:-① Write a equivalen NFA for the following NFA-ϵ



Sol:-

ϵ-closure $(q_0) = \{q_1, q_2, q_0\}$

ϵ-closure $(q_1) = \{q_1, q_2\}$

ϵ-closure $(q_2) = \{q_2\}$

NFA:- Starting State is q_0

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

No of states in NFA is same as the no of States in NFA-ϵ Moves.

The transistion function δ' is as follows.

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

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

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

$= \epsilon$-closure $(q_0, \phi, \phi\}$

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

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

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

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

$= \epsilon$-closure $(\phi \cup q_1 \cup \phi)$

$= \epsilon$-closure (q_1)

$= \{q_1, q_2\}$

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

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

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

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

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

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

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

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

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

$= \phi$

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

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

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

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

$= \{q_1,q_2\}$

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

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

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

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

$= \{q_2\}$

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

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

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

$= \epsilon\text{-closure}\,\phi$

$= \phi$

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

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

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

$= \phi$

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

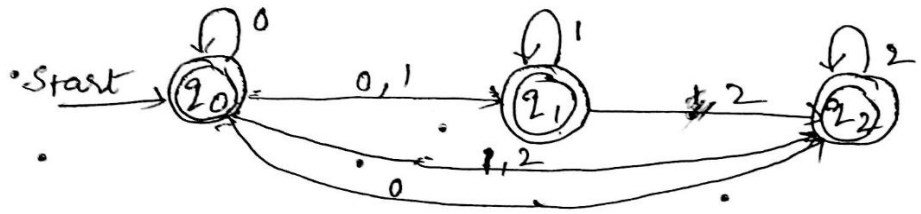
$= \epsilon\text{-closure}(\delta(q_2, 2))$

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

- The final states $\{F'\}$ for NFA can be obtained by observing the ϵ closures of q_0, q_1 and q_2.
- If the final state q_2 is present in the ϵ-closure of a particular state, then that state is a final state of NFA.
- Heare q_2 is present in ϵ-closure(q_0) and ϵ-closure(q_1) and ϵ-closure(q_2)

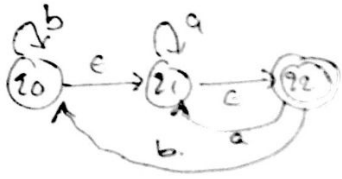
$\therefore \{F\} = \{q_0, q_1, q_2\}$

$\therefore$ The N.F.A is as follows



States	0	1	2
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	$\{q_2\}$

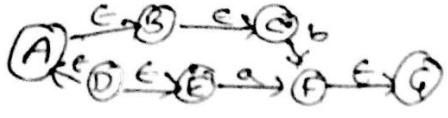
Ex:



NFA for given NFA-ϵ Moves is

$M' = (Q = \{q_0, q_1, q_2\}, \Sigma' = \{0, 1, 2\}, \delta', q_0, F' = \{q_0, q_1, q_2\})$

1- Find ϵ-closure(A) and ϵ-closure(C) for finite Automata



$\epsilon\text{-cl}(A) = \{A, B, C\}$

$\epsilon\text{-cl}(C) = \{C\}$.

obtained

Regular Expressions :-

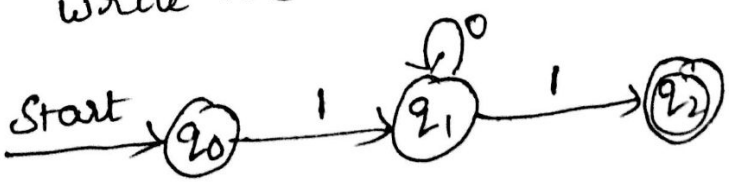
- Any boolean expression or simple expression that is used to represent a F.A is called regular expression. (or)
- A language accepted by finite automata are easily described by simple expressions called Regular expressions.

Ex:- ϕ, ϵ are regular expressions.

- Let Σ be an alphabet. The regular expression over Σ and the sets that they denote are defined recursively as follows.

1) ϕ is a regular expression and denotes the empty set
2) ϵ is a regular expression and denotes the set $\{\epsilon\}$
3) for each a in Σ, a is a regular expression and denotes the set $\{a\}$
4) If r and s are regular expressions denoting language R and S, respectively then $(r+s)$, (rs) and (r^*) are regular expressions that denote the sets $R \cup S$, RS and R^* Respectively.

Ex:- write the R.E for the following finite automata (By default N.F.A ϵ Moves)



R.E = 1 0* 1

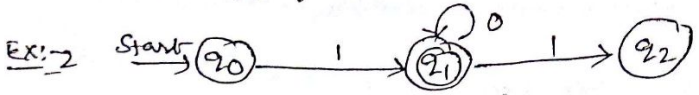
= 1, {∈, 0, 00, 000 ---}, 1

= {1, 10, 100, 1000 --- }

= {11, 101, 1001, 10001 --- }

∴ 10* 1 is the corresponding R.E.

Ex:-2



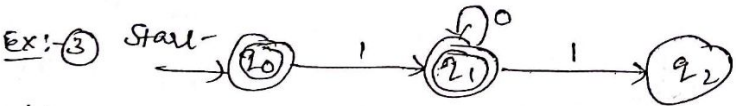
Sol: valid strings generated by the F.A are

1, 10, 100, 1000 ---

R.E = 1 0*

= 1 {∈, 0, 00, 000 ---} = {1, 10, 100, 1000 ---}

Ex:-(3)



Sol: valid strings generated by give FA are

∈, 1, 10, 100, 1000 ---

RE = ∈ + 10*

= ∈ + 1 {∈, 0, 00 ---}

= ∈ + {1, 10, 100 ---- } = {∈, 1, 10, 100, 1000 ---}

Ex:-(4) Construct a R.E for the following language L where L = {abb, a, bba}

Sol:- R.E = abb + a + bba

Ex:-(5) Construct a R.E for the L which has at least two 0's over Σ = {0, 1}

Sol:- R.E = (1+0)* 00 (1+0)*

Ex:-6:- write a R.E for language accepting all combinations over the set Σ = {a} Ans:- R = a*

Ex:-7:- write a R.E for language containing any no of a's and followed by any no of b's

R.E = (a+b)*

Ex:-8:- Construct R.E for language accepting all the strings which are starting with 1 and ending with '0' Ans:- R.E = 1 (0+1)* 0

Ex:-9:- Strings which are ending with 00 over Σ = {a, b} Ans:- R.E = (0+1)* 00

Ex:- write a R.E to denote language L over $\Sigma = \{a, b, c\}$ in which every string will be such that any number of a's followed by any no of b's followed by any no of c's

Ans:- R.E = $a^* b^* c^*$

Ex:- Construct a R.E for language L which accepts all the strings over with atleast two b's over $\Sigma = \{a, b\}$

Sol:- $R = (a+b)^* b (a+b)^* b (a+b)^*$

Ex:- Construct R.E for the language which consists of exactly two b's over the set $\Sigma = \{a, b\}$

R.E:- $a^* b a^* b a^*$

Ex:- write a R.E to denote a language L over $\Sigma = \{a, b\}$ such that the 3rd character from right end of the string is always 'a'.

R.E:-

R.E :- [anyno of chactn of a's and b's] [a] [either a or b] [either a or b]

3rd 2nd 1st

$R.E = (a+b)^* a (a+b) (a+b)$

Ex - Construct R.E which denotes a language L over the set $\Sigma = \{0\}$ having even length of string.

Sol :- $\Sigma = \{0\}$ there are strings in L of even length

i.e $L = \{\epsilon, 00, 0000, 000000 \cdots\}$

$\therefore R.E = (00)^*$

Ex Write a R.E which denotes a language L over set $\Sigma = \{1\}$ having odd length of strings.

Sol :- $R = 1(11)^*$

$= L = 1\{\epsilon, 11, 1111 \cdots\}$

$L = \{1, 111, 11111 \cdots\}$

Ex:- Write a R.E to denote the language L over $\Sigma = \{a, b\}$ such that all the strings do not contain the substring "ab".

$L = \{\epsilon, a, b, bb, aa, ba \cdots\}$

R:- $(b^* a^*)$

Show that $L(a^*b^*) \neq L(ab)^*$

$L(a^*b^*) = \{\epsilon, a, aa, aaa, \ldots\} \{\epsilon, b, bb, bbb, \ldots\}$

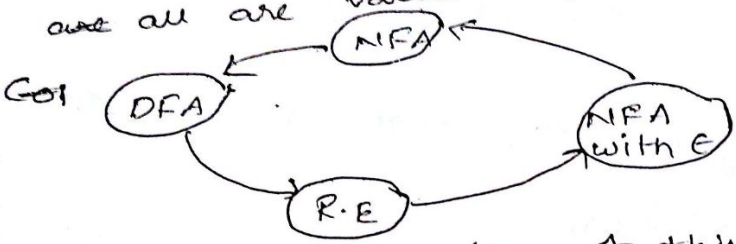
$= \{\epsilon, b, bb, bbb, \ldots a, ab, abb, \ldots aa, aab, aabb \ldots\}$

$L(ab)^* = \{\epsilon, ab, abab, ababab \ldots\}$

$\therefore L(a^*b^*) \neq L(ab)^*$

Regular Set:- A language is denoted by a regular expression is said to be a regular set. (or) A regular set a set consisting of valid regular expression. It is usually denoted by capital letters.

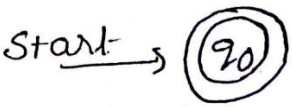
* R, S be a regular set then R.S, R∪S, R^*, S^* ... are all are valid regular sets.



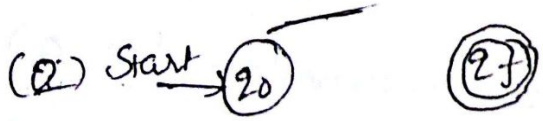
Construction of this chapter

Conversion of regular expression to NFA

(1) For ϵ, Construct the NFA

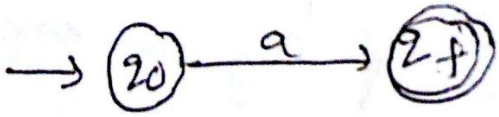


$r = \epsilon$

(2) 

$r = \phi$

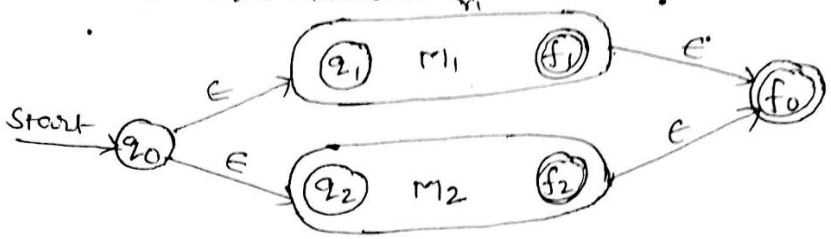
(3) For a in Σ, construct NFA



$r = a$

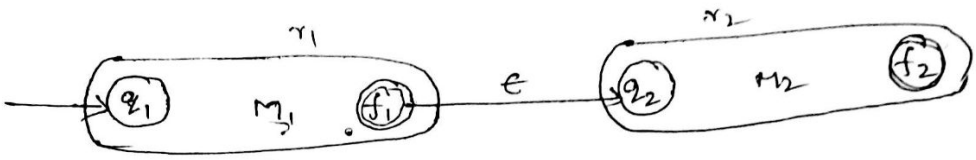
(4) Suppose r_1 and r_2 are any two R.E generated by Machines M_1 and M_2 respectively.

(i). Then the machine for $r_1 + r_2$ (or) $r_1 \cup r_2$ is follows



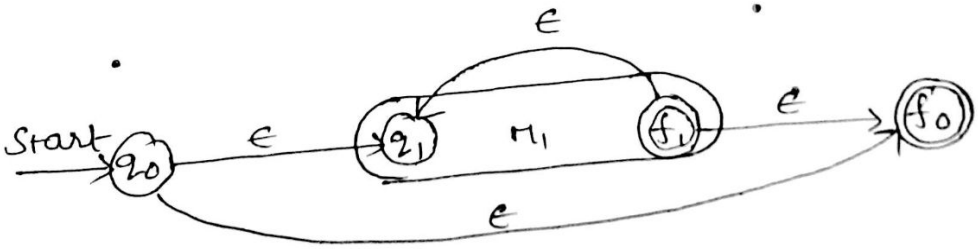
$r_1 + r_2$ (or) $r_1 \cup r_2$ (union)

(ii) $r_1 \cdot r_2$ also a R.E generated by $M_1 M_2$



$r_1 \cdot r_2$ (concatenation)

(iii) For the regular expression r^* is generated by a Machine M_1



r_1^* (closure)

Algebraic Law for R.E :-

- Two regular expressions P and Q are equivalent (i.e P = Q) if and only if P represents the same set of string as Q does. For showing this equivalance of regular expression we need to show some identities of Regular expressions

Algebraic Laws for RE :-

Let P, Q, R are regular expressions then the identity rules are as given below.

1. $\epsilon R = R\epsilon = R$
2. $\epsilon^* = \epsilon$
3. $(\phi)^* = \epsilon$
4. $\phi R = R\phi = \phi$
5. $\phi + R = R$
6. $R + R = R$
7. $RR^* = R^*R = R^+$
8. $(R^*)^* = R^*$
9. $\epsilon + RR^* = R^*$
10. $(P+Q)R = PR + QR$
11. $(P+Q)^* = (P^*Q^*) = (P^* + Q^*)^*$
12. $R^*(\epsilon + R) = (\epsilon + R)R^* = R^*$
13. $(R+\epsilon)^* = R^*$
14. $\epsilon + R^* = R^*$
15. $(PQ)^*P = P(QP)^*$
16. $R^*R + R = R^*R$

Ardens theorem :-

$R = Q + RP$ is equalent to $R = QP^*$.

Ex:- Prove $\epsilon + 1^*(011)^*(1^*(011)^*)^* = (1+011)^*$

Ex:- $= \epsilon + 1^*(011)^*(1^*(011)^*)^*$

$\therefore \epsilon + P_1 P_1^*$ $\quad (\because P_1 = 1^*(011)^*)$

$= P_1^*$

$= (1^*(011)^*)^*$

$= (P_2^* P_3^*)^*$ $\quad (\because P_2 = 1 \text{ and } P_3 = (011))$

$= ((P_2 + P_3)^*)^* \Rightarrow (P_2 + P_3)^*$ $\quad [\because (P^* + Q^*)^* = (P+Q)^*) \quad (P^*)^* = P^*$

$= (1+011)^*$

L.H.S = R.H.S

Prove $(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1) = 0^*1(0+10^*1)^*$

Solve L.H.S first-

$= (1+00^*1)(\epsilon + (0+10^*1)^*(0+10^*1)$

$= \epsilon(1+00^*1)\cdot(0+10^*1)^*$ $(\because \epsilon + R^*R = R^*)$

= Taking 1 as common factor

$= (\epsilon+00^*)1(0+10^*1)^*$

Applying $\epsilon+00^* = 0^*$

$= 0^*1(0+10^*1)^*$ $(\because \epsilon+00^* = 0^*)$

= R.H.S

IMS construct 1+0

- Find Regular Expression for DFA as shown

Eqn:- $A = Ab+Bb+\epsilon$ ∴ Start state ——(1)

$B = Aa$ ——(2)

$C = Ba+C(a+b)$ ——(3)

We will Solve eqn(1) by putting eqn(2) in it.

$A = Ab+Aab+\epsilon \Rightarrow A(b+ab)+\epsilon$

R P Q

$\therefore A = \epsilon(b+ab)^*$ $(R = QP^*$ Arden's theorem)

$A = (b+ab)^*$ $\because R^*\epsilon = R^*$ ——(4).

Put eqn(4) in eqn(2) we will get

$B = Aa \Rightarrow (b+ab)^*a$

Put value of B in eqn(3), we will then get.

$C = (b+ab)^*aa+C(a+b)$

Q R P

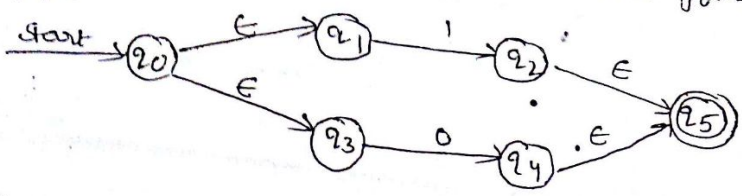
$C = (b+ab)^*aa(a+b)^*$ $\therefore R = Q+RP$ then $R = QP^*$

As C is a final state the regular equation is the equation of state C.

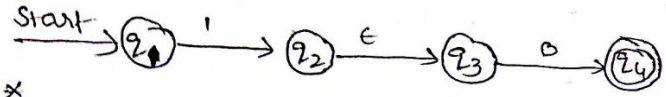
$\therefore e = (b+ab)^*aa(a+b)^*$.

Construct the NFA with ϵ Moves for the following R.E's

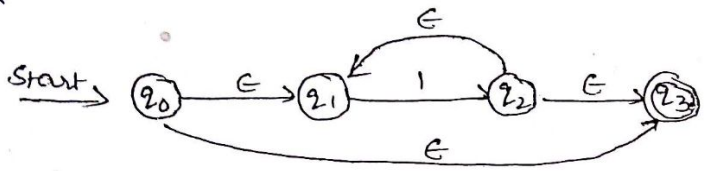
(i) 1+0



(ii) 1.0



(iii) 1^*



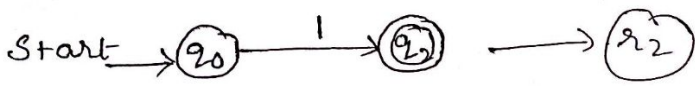
} * conca +

Ex:- Construct the NFA for the R.E 01^*+1

Sol:- $r_1 + r_2 = 01^* + 1$

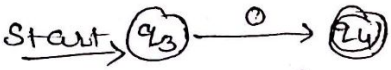
$r_1 = 01^* \quad r_2 = 1$

(i) The automation for r_2 is

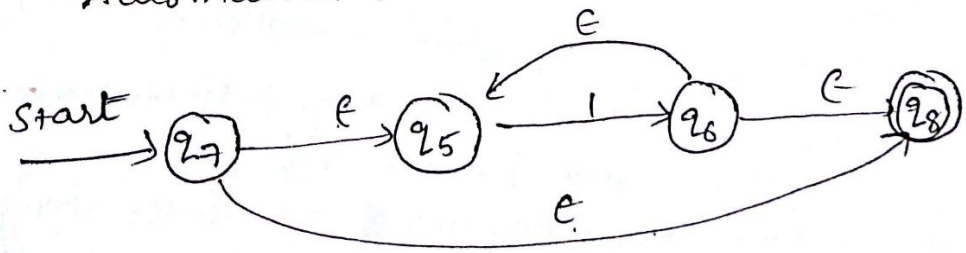


(ii) we may express r_1 as $r_3 r_4$

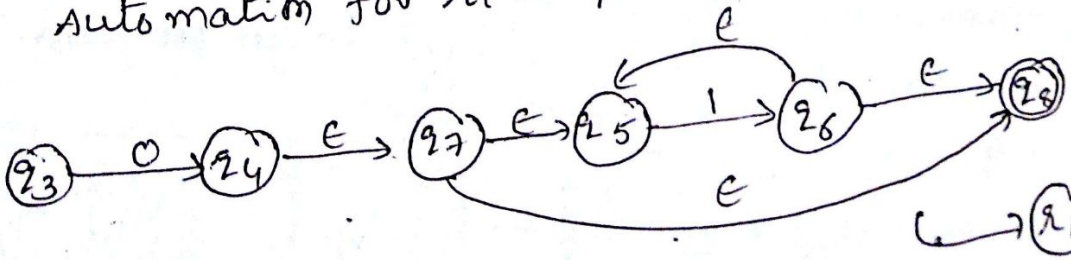
$\therefore r_1 = 01^* \quad r_3 = 0$ and $r_4 = 1^*$

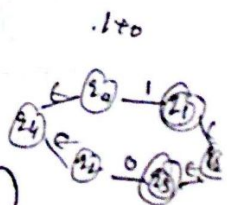
so r_3 is 

(iii) Automation for r_4 is 1^*



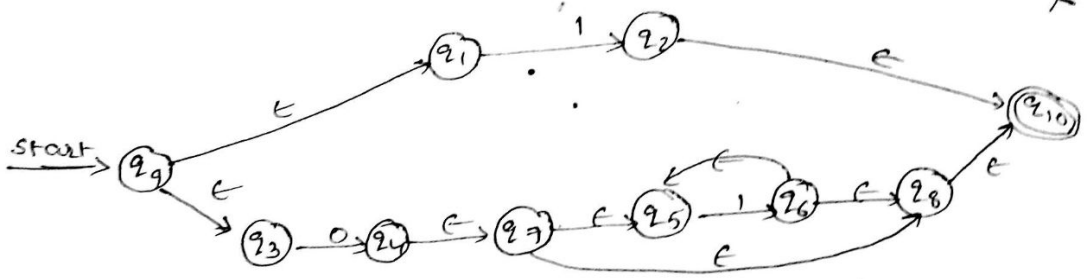
(iv) Automation for r_1 is 01^*





18 19

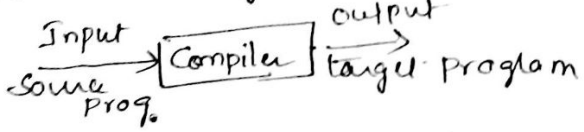
(v) final automaton NFA for $r_1 + r_2$ is



$r = 01^* + 1$

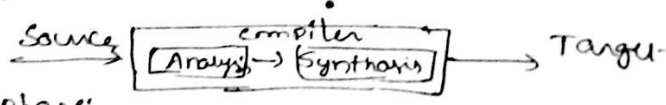
Compilers:-

- A compiler is a program that reads a program written in one language - the source language - and translates it into an equivalent program in another language - the target.



Compilation can be done in two parts

(i) Analysis phase (ii) Synthasis phase.



(i) Analysis phase:-

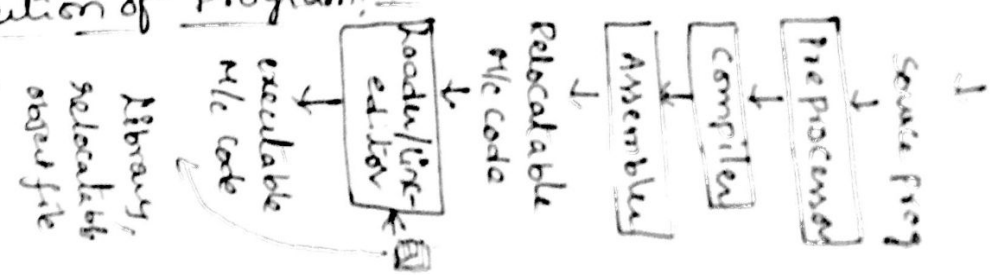
(i) lexical analysis:- Source program is broken down into tokens

(ii) Syntax Analysis:- The tokens are arranged in hierarchical structure that helps findings the Syntax of source string

(iii) Semantic analysis:- Real meaning of the source string determine

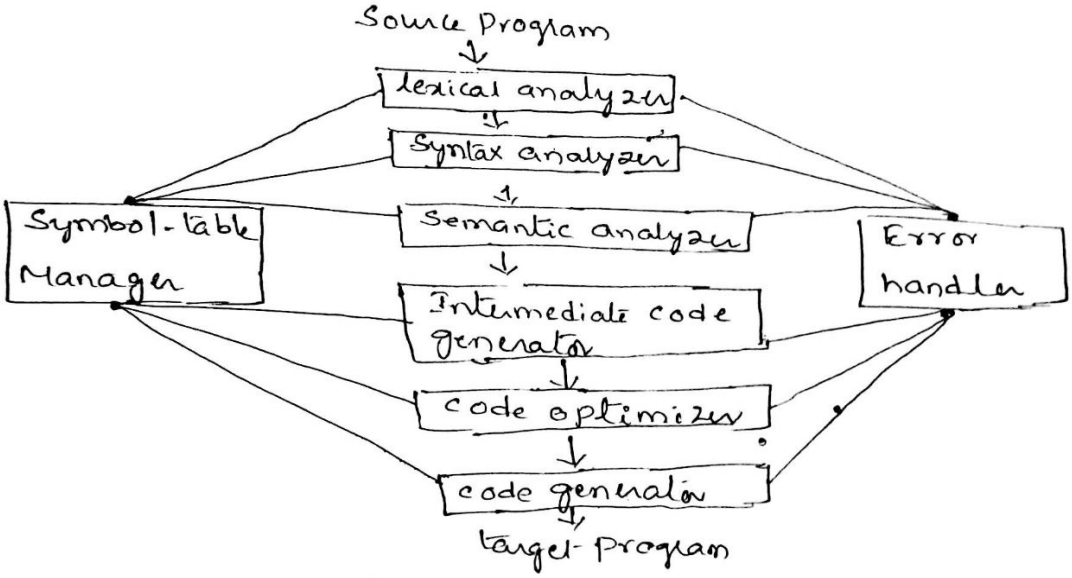
(ii) Synthasis phase: Intermediate form of source language is taken and converted into an equivalent target program.

Process of execution of program:-



The Phases of a Compiler :-

- A compiler operates in phases, each of which transforms the source program from one representation to another.
- A typical decomposition of a compiler as shown bellow



(1) Lexical analysis :- In which the stream of characters making up the source program is read from left to right and grouped into tokens that are sequences of characters having a collective meaning.

- In a compiler, linear analysis is called lexical analysis (or) scanning. In lexical analysis the characters in the assignment

position := initial + rate * 60

would be grouped into the following tokens

1. the identifier position
2. assignment symbol :=
3. The identifier initial
4. plus sign
5. The identifier rate
6. The multification sign
7. The number 60

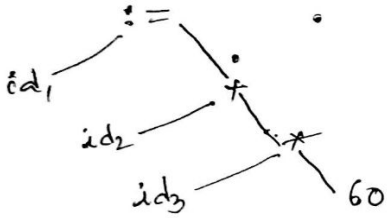
- The blanks separating the characters of these tokens would normally be eliminated during lexical analysis.

Semantic Analysis :-

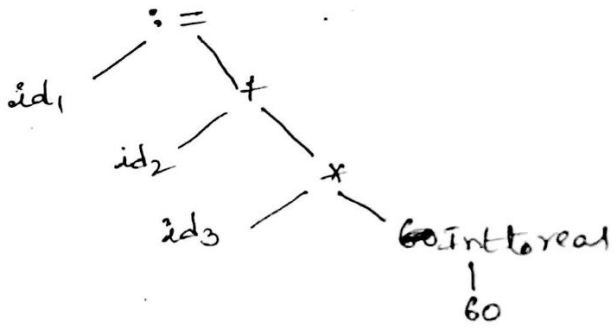
- Semantic analysis phase checks the Source Program for semantic errors and gathers type information for the subsequent code generation phase.
- Important component of Semantic analysis is type checking Hear the compiler checks that each operator has operands that are permitted by the source language specification.

Syntax Analysis :- The Syntax analysis is also called Parsing. In this phase the tokens generated by lexical analyzer are grouped together to form a hierarchical structure.

- The hierarchical structure in this phase is called Parse tree (or) Syntax tree.



Syntax analyzer



Intermediate code generation :-

- Intermediate code is a kind of code which is easy to generate and this code can be easily converted to target co

```
temp1 := inttoreal(60)
temp2 := id3 * temp(1)
temp3 := id2 + temp2
id1   := temp3
```

Code optimization :- This phase attempts to improve the intermediate code. This is necessasary to have faster execute code (or) less consumption of memory. So overall running tim of target Program can be improved.

```
temp1 := id3 * 60.0
id1   := id2 + temp1.
```

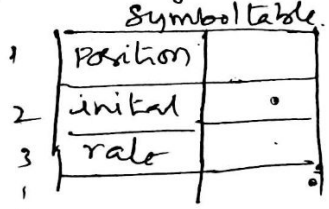
Code generation:-

- Code generation phase the target code gets generated. The intermediate code instruction are translated into sequence of M/c instructions.

```
MOVF  id3, R2
MULF  #60.0, R2
MOVF  id2, R1
ADDF  R2, R1
MOVF  R1, id1
```

Symbol table Management:-

- To support these phases of compiler a symbol table is maintained. The task of symbol table is to store identifiers used in program
- Basically symbol table is a data structure used to store information about identifiers.
- Symbol table also stores information about attributes of each identifier. attributes of identifiers are usually its type and scope and information.



Error handler:- As program written by human beings. therefore they can not be free from errors. In compilation each phase detect errors. These errors must be reported to error handler whose task is handle the errors so that compilation can processes

- Normally errors are reported in the form of message.
- When the i/p characters from the i/p do not form the token the lexical analyzer detects it as error.
- Large number of errors can be detected in Syntax analysis phase. Such errors popularly called Syntax error.
- During Semantic type mismatch kind of error is usually detected.