Unit = 2

Subject Code: KCS402

Section: A & B

Regular Expressions & Languages:-

Regular Expressions:-

The regular expressions are useful for representing certain sets of strings in an algebric fashion.

Regular expressions over Σ as follows:

1) Any terminal symbol (i.e an element of Σ), Λ and ϕ are regular expressions. When we view a in Σ as a regular expression, we denote it by a.

2) The union of two Regular expressions R_1 & R_2, written as $R_1 + R_2$ is also a regular expression.

3) The concatenation of two regular expressions R_1 & R_2, written as $R_1 R_2$, is also a regular expression.

4) The iteration (or closure) of a regular expression R, written as R^*, is also a regular expression.

5) If R is regular expression, then (R) is also a regular expression.

In the absence of parentheses, for evaluating a regular expression involving various operation, we perform iteration first, then concatenation, & finally union.

Any set represented by a regular expression is called a regular set.

if, for example, $a, b \in \Sigma$, then (i) a denotes the set $\{a\}$, (ii) a+b denotes $\{a, b\}$ (iii) ab denotes $\{ab\}$ (iv) a^* denotes the set $\{\Lambda, a, aa, aaa, \ldots\}$ (v) $(a+b)^*$ denotes $\{a, b\}^*$.

The set represented by R is denoted by L(R).

Let R_1 & R_2 denote any two regular expressions. Then

(i) a string in $L(R_1+R_2)$ is a string from R_1 or string from R_2

(ii) a string in $L(R_1R_2)$ is a string from R_1 followed by string R_2 from R_2.

Consequently

(i) the set represented by R_1+R_2 is the union of the sets represented by R_1 & R_2

(ii) the sets represented by R_1R_2 is the concatenation of sets R_1 & R_2

example:-

Describe following set by regular expressions:-

(a) $\{101\}$:- 101 is obtained by concatenating 1, 0 and 1
So $\{101\}$ is repr. by 101

(b) $\{01, 10\}$:- As it is union of $\{01\}$ & $\{10\}$, we have $\{01, 10\}$ rep. by 01+10

(c) $\{\Lambda, 0, 00, 000, \ldots\}$:- is simply $\{0\}^*$, it is represented by 0^*

(d) $\{1, 11, 111, \ldots\}$:- 1 is any an element of $\{1\}^*$. Hence $1(1)^*$ represents $\{1, 11, 111, \ldots\}$

example: L_1 = set of all strings of 0's & 1's ending in 00
$= (0+1)^*00$.

L_2 = Set of all string of 0's & 1's beginning with 0 & ending with 1.
$0(0+1)^*1$

$L_3 = \{\Lambda, 11, 1111, 111111, \ldots\}$
$= (11)^*$

IDENTITIES for REGULAR EXPRESSIONS :-

Subject Code: 1CS-402

(2)

Section: A & B

(I1) $\phi + R = R$ (I2) $R + R = R$ (I3) $RR^* = R^*R$

(I4) $\phi R = R\phi = \phi$ (I5) $R^*R^* = R^*$ (I6) $(R^*)^* = R^*$

(I7) $\Lambda + RR^* = \Lambda + R^*R$ (I8) $(PQ)^*P = P(QP)^*$

(I9) $(P+Q)R = PR + QR$ (I10) $\Lambda + RR^* = R^* = \Lambda + R^*R$

& $R(P+Q) = RP + RQ$

Arden's Theorem :-

Let P & Q be two regular expressions over Σ. if P does not contain Λ, then the following equation in R, namely

$$R = Q + RP \quad \longrightarrow (5.1)$$

has a unique solution (i.e one & only one solution) given by $R = QP^*$

Proof: $Q + RP = Q + (QP^*)P = Q(\Lambda + P^*P) = QP^*$

Hence (5.1) is satisfied when $R = QP^*$. This means $R = QP^*$ is a solution of (5.1)

to prove uniqueness, Here replacing R by $Q + RP$ on R.H.S we get

$$\begin{aligned} Q + RP &= Q + (Q + RP)P \\ &= Q + QP + RPP \\ &= Q + QP + RP^2 \\ &= Q + QP + (Q + RP)P^2 \\ &= Q + QP + QP^2 + \cdots + QP^i + RP^{i+1} \\ &= Q(\Lambda + P + P^2 + \cdots + P^i) + RP^{i+1} \end{aligned}$$

from (5.1)

$$R = Q(\Lambda + P + P^2 + \cdots + P^i) + RP^{i+1} \quad \text{for } i \geq 0$$

We know show that any soln of (5.1) is equivalent to QP^*. Suppose R satisfies (5.1), then it satisfies (5.2)

Prove that regular expression

eg:- $R = \Lambda + 1^*(011)^*(1^*(011)^*)^*$

also describes the same set of strings.

Soln:-

$R = \Lambda + P_1 P_1^*$, where $P_1 = 1^*(011)^*$

$= P_1^*$ (using $\Lambda + RR^* = R^* = \Lambda + R^*R$)

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

$= (P_2^* P_3^*)^*$ letting $P_2 = 1$, $P_3 = 011$

$= (P_2 + P_3)^*$ using $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$

$= (1 + 011)^*$

Finite Automata and Regular Expressions :-

The transition systems can be generalized by permitting Λ-transitions or Λ-moves which are associated with a null symbol Λ.

Suppose we want to replace a Λ move from vertex V_1 to vertex V_2. Then

Step 1:- Find all the edges starting from V_2

step 2 :- Duplicate all these edges starting from V_1, without changing the edge labels

~~step 3~~:-

step 3:- if V_1 is an initial state, make V_2 also as initial state

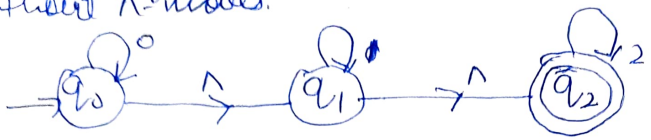
Step 4:- if V_2 is a final state, make V_1 also as final state

Subject Code: KCS 402 ③

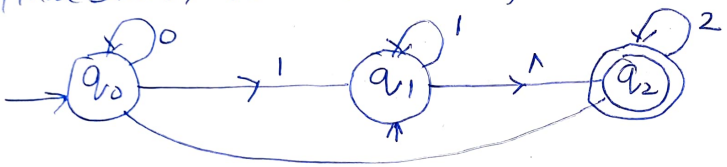
Manav Rachna International Institute of Engineering & Technology ★ Greater Noida ★ Lecture No.

Section: A1 B

eg:- Consider a finite automaton with $\wedge$-moves, given in fig. Obtain an equivalent automaton without $\wedge$-moves.

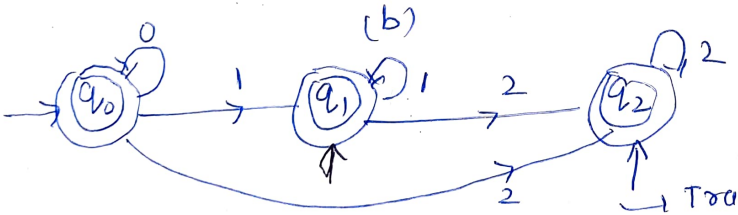


sol:- We first eliminate $\wedge$-move from q_0 to q_1 to get fig. 5.2(a). q_1 is made an initial state. Thenwe eliminate $\wedge$ move from q_0 to q_2 in 5.2(a) to get 5.2 (b). As q_2 is final state, q_0 is also made final state.



(a)

(b)



→ Transition System without $\wedge$-moves.

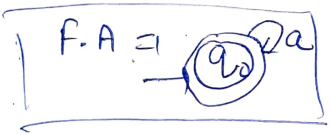
Kleene's Theorem :-

(1) For any regular expression 'r' that represents language L(r) there is a finite automaton that accept same language.

(2) For any FA 'M' that accepts language L(M), there is a regular expression that represents the same language.

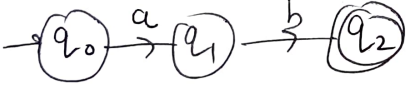
$a^* \rightarrow$ R.E $\Leftarrow$ Language L is accepting a^* $\Leftarrow$ $L = \{a^*\}$ over $\Sigma = \{a\}$

Part 1 :-



Part 2 :-

$L = \{ab\}$ $\Rightarrow$ Machine



This Machine is accepting ab.

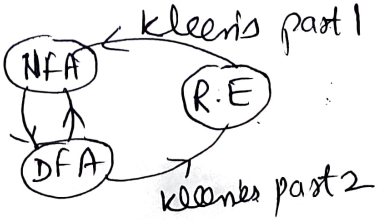
We can makes Regular expression of ab which represent that language.

Proof :-

Part 1 :-

Graphically :-

Part 1 ✓



Subject Code: KOS-402 (4)

Lecture No.

Section: A & B

Proof for Part 1:

For any R.E we show how to construct an equivalent NFA.

Induction Method:

Basic step:

Minimum Regular expression: (i) ϕ, $\in$, a

Equivalent Machine → ○ → ◎ ○ —a→ ◎

(ii) Union, concatenation, *

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

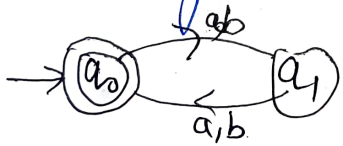
if (a+b) is R.E then we can make NFA of that.

if (ab) " " " " " "

$(ab)^*$ " " " " " "

Part 2:

DFA Accepting even No. of strings over (a,b)



L = Even no's of string over {a,b}

Equivalent R.E of this is

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

Algebraic Method Using Arden's Theorem:

The following Method is an extension of Arden's theorem. This is used to find r.e recognized by a transition system.

The following assumptions are made regarding the transition system.

(i) transition graph does not have $\wedge$-moves.

(ii) it has only one initial state, V_1.

(iii) its vertices are $V_1 - V_n$.

(iv) V_i the r.e represents the set of strings accepted by the system even though V_i is a final state.

(v) set of eqn in $V_1 \text{ —— } V_n$.

$$V_1 = V_1\alpha_{11} + V_2\alpha_{21} + \text{——} V_n\alpha_{n1} + \wedge$$

$$V_2 = V_1\alpha_{12} + V_2\alpha_{22} + \text{——} + V_n\alpha_{n2}$$

$$\vdots$$

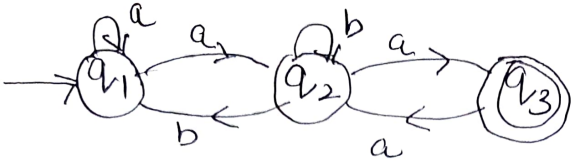
$$V_n = V_1\alpha_{1n} + V_2\alpha_{2n} + \text{——} V_n\alpha_{nn}$$

For getting set of strings recognized by transition system, we have to take 'union' of all V_i's corresponding to final state.

Example:

Consider the transition system given. Prove that the strings recognized are

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



Subject Code: KAS-402 (5)

Lecture No.

Section: A & B

Soln:- We can directly apply above method since the graph does not contain any $\wedge$-move and there is only one initial state

The three equations for $q_1, q_2, \& q_3$ can be written as

$$q_1 = q_1a + q_2b + \wedge \qquad q_2 = q_1a + q_2b + q_3a \qquad q_3 = q_2a$$

It is necessary to reduce number of unknowns by repeated substitution. By substituting q_3 in the q_2 eqn. we get by applying Theorem (Arden's)

$$q_2 = q_1a + q_2b + q_2aa$$
$$= q_1a + q_2(b + aa)$$
$$= q_1a(b + aa)^*$$

Using $R = Q + RP$, $R = QP^*$

Substituting q_2 in q_1 we get

$$q_1 = q_1a + q_1a(b + aa)^*b + \wedge$$
$$= q_1(a + a(b + aa)^*b) + \wedge$$

Hence $q_1 = \wedge(a + a(b + aa)^*b)^*$

$q_2 = (a + a(b + aa)^*b)^* a(b + aa)^*$

$q_3 = (a + a(b + aa)^*b)^* a(b + aa)^*a$

Since q_3 is final state, set of strings recognized by graph is

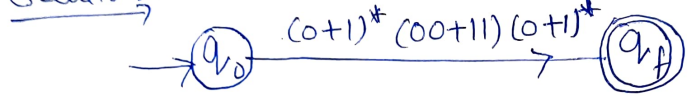
$(a + a(b + aa)^*b)^* a(b + aa)^*a$

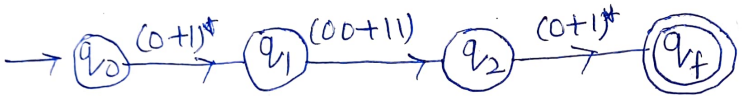
Construction of Finite Automata Equivalent to a Regular Expressions:-

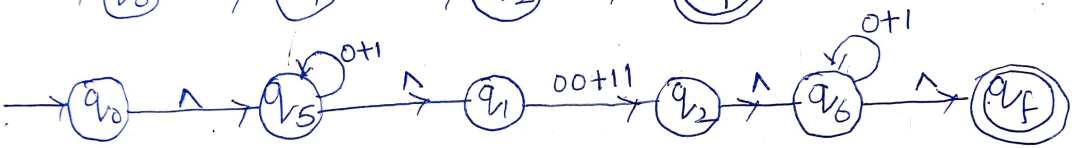
Construct the finite automaton equivalent to the regular expression

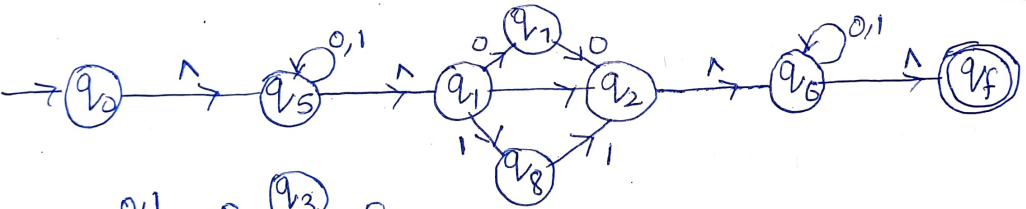
$(0+1)^*(00+11)(0+1)^*$

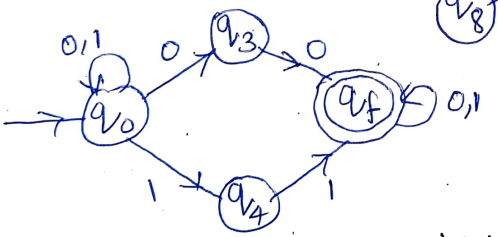
Solution:-











Step 2:- (Construction of DFA) We construct the transition table for the NDFA

Transition table

State / Σ	0	1
→ q_0	q_0, q_3	q_0, q_4
q_3	q_f	
q_4		q_f
(q_f)	q_f	q_f

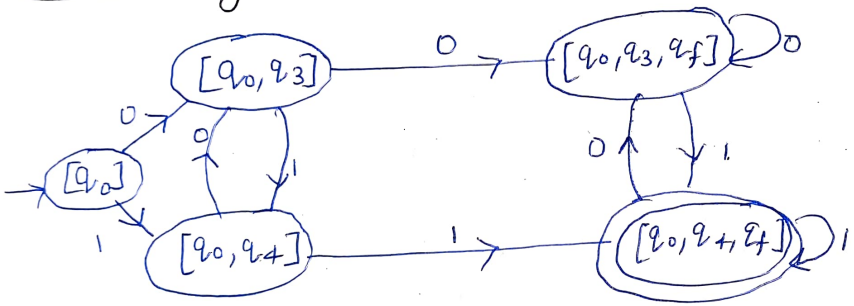
Subject Code: KCS-402

Section: A & B

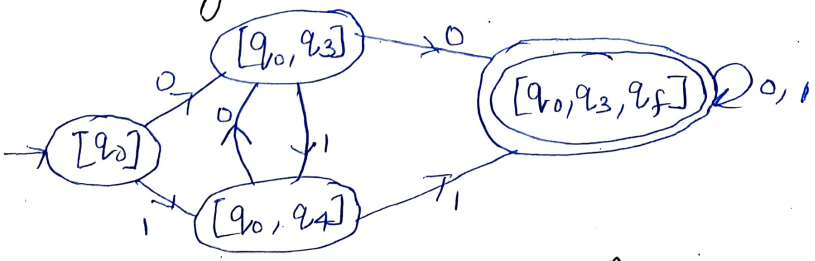
Transition Table for DFA :-

Q.	Q_0	Q_1
→ $[q_0]$	$[q_0, q_3]$	$[q_0, q_4]$
$[q_0, q_3]$	$[q_0, q_3, q_f]$	$[q_0, q_4]$
$[q_0, q_4]$	$[q_0, q_3]$	$[q_0, q_4, q_f]$
$[q_0, q_3, q_f]$	$[q_0, q_3, q_f]$	$[q_0, q_4, q_f]$
$[q_0, q_4, q_f]$	$[q_0, q_3, q_f]$	$[q_0, q_4, q_f]$

The State diagram for the successor table is



Finally, we try to reduce the number of states (This is possible when two rows are identical in the successor table) As the rows corresponding to $[q_0, q_3, q_f]$ and $[q_0, q_4, q_f]$ are identical, we identify them. The state diagram for equivalent automaton, where the number of states is reduced is described by

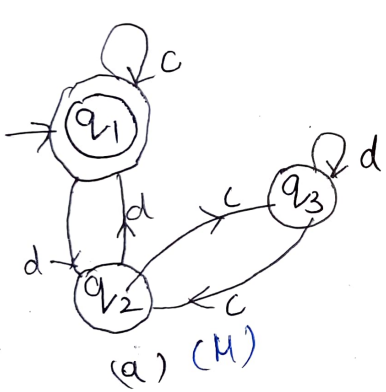


Reduced Finite Automaton

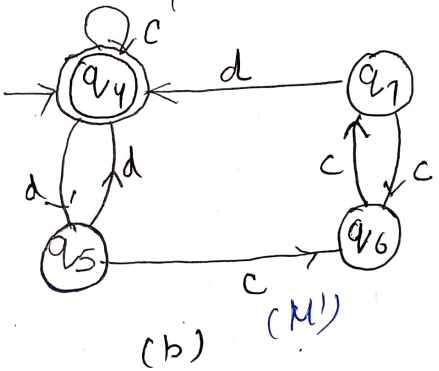
Equivalence of Two Finite Automata:-

Two finite automata over Σ are equivalent if they accept the same set of strings over Σ. When two finite automata are not equivalent, there is some string w over Σ satisfying the following: one automaton reaches a final state on application of w, whereas the other automaton reaches a non-final state.

Consider the following two DFA's M and M' over $\{0,1\}$. Determine whether M & M' are equivalent.



(a) (M)



(b) (M')

Solution:

The initial states in M & M' are q_1 & q_4. Hence the first element of first column in the comparison table must be (q_1, q_4). The first element in the second column is (q_1, q_4) since both q_1 & q_4 are c-reachable from respective initial states

Section:.....

Comparison table

II)

(q, q')	(q_c, q'_c)	(q_d, q'_d)
(q_1, q_4)	(q_1, q_4)	(q_2, q_5)
(q_2, q_5)	(q_3, q_6)	(q_1, q_4)
(q_3, q_6)	(q_2, q_7)	(q_3, q_6)
(q_2, q_7)	(q_3, q_6)	(q_1, q_4)

As we do not not get a pair (q, q'), where q is a final state & q' is a non final state, we proceed until all the elements in the second & third columns are also in the first column. Therefor M & M' are equivalen

Pigeonhole Principle:-

Proof:- (1) It states that "if n pigeons are assigned to m pigeonholes then atleast one pigeonhole contains two or more pigeons ($m<n$)".

(2) The pigeonhole principle is sometime useful in counting methods.

Proof:- (1) Let m pigeonholes be numbered with the number 1 through m.

(2) Beginning with pigeon 1, each pigeon is assigned in order to the pigeonholes with the same number.

(3) Since $m<n$ ie the number of pigeonhole is less than the number of pigeons, $n-m$ pigeons are left without having assigned a pigeonhole.

(4) Thus, at least one pigeonhole will be assigned to a more than one pigeon.

(5) We note that the pigeonhole principle tells us nothing about how to locate the pigeonhole that contains two or more pigeons.

(6) It only assests the existence of a pigeon hole containing two or more pigeons.

(7) To apply the principle one has to decide which objects will play the role of pigeon & which objects will play the role of pigeonholes.

Subject Code: KCS-402

Section: A-2B

State & prove the Pumping lemma for regular languages

In pumping lemma we give a necessary condition for an input string to belong to a regular set. The result is called pumping lemma as it gives a method of pumping (generating) many input strings from a given string. As pumping lemma gives a necessary condition, it can be used to show that certain sets are not regular.

Theorem (Pumping lemma):-

Let $M=(Q, \Sigma, \delta, q_0, F)$ be a finite automaton with n states. Let L be the regular set accepted by M. Let $w \in L$ & $|w| \geq m$. if $m \geq n$, then there exists x, y, z such that $w = xyz$, $y \neq \wedge$ & $xy^i z \in L$ for each $i \geq 0$

Proof:- Let $w = a_1 a_2 \cdots a_m$, $m \geq n$

$\delta(q_0, a_1 a_2 \cdots a_i) = q_i$ for $i = 1, 2, \ldots, m$; $Q_1 = \{q_0, q_1 \cdots q_m\}$

That is, Q_1 is the sequence of states in the path with path value $w = a_1 a_2 \cdots a_m$. As there are only n distinct states atleast two states in Q_1 must coincide.

Among the various pairs of repeated states, we take the first pair. Let us take them as q_j & q_j ($q_j = q_k$).

Then j & k satisfy the condition $0 \leq j < k \leq n$.

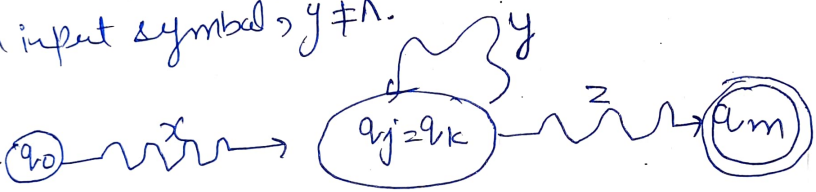
The string w can be decomposed into three substrings $a_1 a_2 \cdots a_j$, $a_{j+1} \cdots a_k$ & $a_{k+1} \cdots a_m$.

Let x, y, z denote these strings $a_1 a_2 \dots a_j$, $a_{j+1} \dots a_k$, $a_{k+1} \dots a_m$, respectively.

As $k \le n$, $|xy| \le n$ & $w = xyz$. The path with path value w in transition diagram of M is shown in below fig.

The automaton M starts from initial state q_0. On applying the string x, it reaches $q_j (= q_k)$: On applying the string y, it comes back to $q_j (= q_k)$. So after application of y^i for each $i \ge 0$, the automaton is in same state q_j.

On applying z, it reaches q_m, a final state. Hence $xy^i z \in L$. As every state in Q_1 is obtained by applying an input symbol, $y \ne \Lambda$.



string accepted by M.

Application of Pumping Lemmas:

This theorem can be used to prove that certain sets are not regular. We now give the steps needed for proving that a given set is not regular.

Step 1:- Assume that L is regular. Let n be the number of states in the corresponding Finite automaton.

Step 2:- Choose a string w such that $|w| \ge n$. Use Pumping lemma to write $w = xyz$, with $|xy| \le n$ & $|y| > 0$.

Subject Code: KCS-402

Step3: Find a suitable integer i such that $xy^i z \notin L$. This contradicts our assumption.

Hence L is not regular.

Decision problem and Decidability properties in regular language:-

(1) A decision problem is a restricted type of an algorithmic problem where for each input there are only two possible outputs.

(2) A decision problem is a function that associates with each input instance of the problem a truth value true or false.

(3) A decision algorithm is an algorithm that computes the correct truth value for each input instance of a decision problem. The algorithm A has to terminate on all inputs.

(4) A decision problem is decidable if there exists a decision algorithm for it. Otherwise it is undecidable.

Decidability properties of regular languages:-

(1) DFA membership :- Let $M = (Q, \Sigma, \delta, q_0, F)$ denote a deterministic finite automaton & x is a sequence of finite-domain variables $(x_1, x_2, \ldots x_n)$ with respective domains $D_1, D_2, \ldots D_n \subseteq \Sigma$. Under a regular langu

membership constraint regular (x, M), any sequence of values taken by the variables of x must belong to the regular language recognized by M.

(2) DFA emptiness :- DFA is said to be empty if there is no path from initial state to final state.

(3) DFA equivalence :- Let $M_1 = (Q, \Sigma, \delta, q_0, F)$ & $M_2 = (Q', \Sigma', \delta', q_0', F')$ are two DFA. M_1 & M_2 are said to be equivalent if and only if $L(M_1) = L(M_2)$.

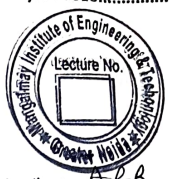
(4) DFA finiteness :- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. String accept by M is said to be finite if and only if there is no cycle between any two state q_1 & q_2.

Closure properties of Regular Languages :-

if you are able to generate F.A of any lang., then that is called R.L.

(1) Union $(L_1 \cup L_2) = r_1 + r_2$
(2) Concatenation $(L_1 \cdot L_2) = r_1 \cdot r_2$ $\quad L_1 = L(r_1)$, $L_2 = L(r_2)$
(3) Closure $(L^*) = r_1^*$
(4) Complementation $\overline{L} = \Sigma^* - L$
(5) Intersection :- $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$
(6) Difference $L_1 - L_2 = L_1 \cap \overline{L_2}$
(7) Reversal $(L)^R$
(8) Homomorphism
(9) Reverse Homomorphism
(10) Substitution
(11) Infinite Union $\rightarrow$ it is not closed

Subject Code: KCS-402



Section: A & B

Example of pumping lemma:-

show that the set $L = \{a^{c^2} \mid c \geq 1\}$ is not regular.

Solution:-

Step 1:- Suppose L is regular. Let n be the number of states in the finite automaton accepting L.

Step 2:- Let $w = a^{n^2}$. Then $|w| = n^2 > n$. By pumping lemma we can write $w = xyz$ with $|xy| \leq n$ and $|y| > 0$.

Step 3:- Consider xy^2z. $|xy^2z| = |x| + 2|y| + |z| > |x| + |y| + |z|$ as $|y| > 0$. This means $n^2 = |xyz| = |x| + |y| + |z| < |xy^2z|$.

As $|xy| \leq n$, we have $|y| \leq n$. Therefore

$$|xy^2z| = |x| + 2|y| + |z| \leq n^2 + n$$

i.e. $$n^2 < |xy^2z| \leq n^2 + n < n^2 + n + n + 1$$

Hence, $|xy^2z|$ strictly lies between n^2 & $(n+1)^2$, but is not equal to any one of them. Thus $|xy^2z|$ is not a perfect square and so $xy^2z \notin L$. But by pumping lemma, $xy^2z \in L$. This is a contradiction.