(UNIT-02) :- (Lecture-01)
Regular Expression & Language

Regular Expression is used to define a regular language into algebraic discription.

The Operators of Regular Expression

There are three operator which are used to represent Regular Expression

1. **Union**: Union of two language L_1 & L_2 denoted by $L_1 \cup L_2$, is the set of string that are in either L_1 or L_2, or both

Eg $L_1 = \{001, 10, 111\}$ & $L_2 \{\epsilon, 001\}$

$L_1 \cup L_2 = \{\epsilon, 001, 10, 111\}$

2. **Concatenation**: The Concatenation of language L_1 & L_2 is represented by $L_1 \cdot L_2$. the Concatenation of two language represented by product of the language.

$L_1 = \{001, 10, 111\}$ & $L_2 \{\epsilon, 001\}$

$L_1 \cdot L_2 = \{001, 10, 111, 001001, 10001, 111001\}$

3. **Closure** (Star, Kleene closure) : The Closure of a language L is denoted by L^* and represents the set of those string that Can be formed by taking any no. of string from L, possible With repetitions & Concatenating all of them

Eg. $L = \{0, 1\}$, the L^* is all string of 0's & 1's including Null String

4. The positive closure of a language L is denoted by L^+ & represent the set of those string that can be formed by taking any no. of string from L, possible with repetitions & concatenating all of them, excluding null string.

$$L^+ = L^* - \epsilon$$

Definition of Regular Expression:

The set of regular expression is define by the following rules:

1. Every latter of Σ can be made into a regular expression, Null string, ϵ itself is a regular expression.
2. if r_1 & r_2 are regular expression then.
   (i) (r_1) (ii) $r_1 . r_2$ (iii) $r_1 + r_2$ (iv) r_1^*
   (v) r_2^* (vi) r_1^+ are also regular expression.
3. Nothing else is regular expression.

Q.1. Write Regular expression over alphabate (a,b,c) Containing at least one a and at least one b.

Solu.

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

Q.2. Write the regular expression for the set of string of 0's & 1's Whose fourth symbole from the right end is 1.

Ans:

r:- $(0+1)^* 1 (0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)$

r → $(0+1)^* 1 (0+1)^3$

Q.3 Write the regular expression for the set of strings of an equal no. of 0's & 1's such that in every prefix, the no. of 0's differs from the the no. of 1's by at most 1.

Ans.

$$r = (01 + 10)^*$$

Q.4. Write a regular expression for the set of string of 0's and 1's not Containing 101 as a substring.

Ans. We can analyse the set of strings then it becomes clear that string may start with 0 & 0's may be repeated, Wherever one is encountered then no sigle 0 must follow it so string are like 0000010, 0000111111, 100100

RE → $(0^* 1^* 00)^* 0^* 1$

Q5- Write the regular expression over alphabate {a,b} for the set of string with even no. of a's followed by odd no. of b's

Sol.

$$(aa)^*(bb)^*b$$

$$L = a^{2n} b^{2m+1} \mid n, m \geq 0$$

Q6- Write the regular Expression for language even length of string on alphabate $\Sigma = \{a,b\}$

RE → $((a+b)(a+b))^*$

or

for odd length

RE → $((a+b)(a+b)^*(a+b))$

Q7 find the RE for the following Language.

(i) L = { length of all string divisible by 3 for $\Sigma = \{a,b\}$ }

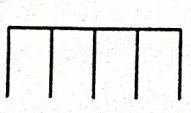
RE → $((a+b)(a+b)(a+b))^*$

(ii) L = { length of all string $\equiv 2 \bmod 3$ for $\Sigma = \{a,b\}$ }

RE → $((a+b)(a+b)(a+b))^*(a+b)(a+b)$

(iii) L = { length of string exactly 2's a for Σ {a,b} }

RE → $b^*ab^*ab^*$



(iv) $L = \{$ a are at least 2 for $\Sigma \{a,b\}\}$

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

(v) $L = \{$ a are at most 2 for $\Sigma = \{a,b\}\}$

RE → $b^* (\in + a)\ b^*\ (\in + a)\ b^*$

Q Write RE over alphabate $\Sigma \{a,b\}$ for the language.

$L = \{$ string starting & ending with same symbole $\}$

RE → $a (a+b)^* a + b (a+b)^* b$

Arden's Theorem:

Let P and Q be two regular expression over alphabate Σ. if P does not contain null string ϵ, then

$$R = Q + RP$$

has a unique solution that is $R = QP^*$

it can be understand as: $R = Q + RP$

Put the value of R in R.H.S.

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

When we put the value again & again

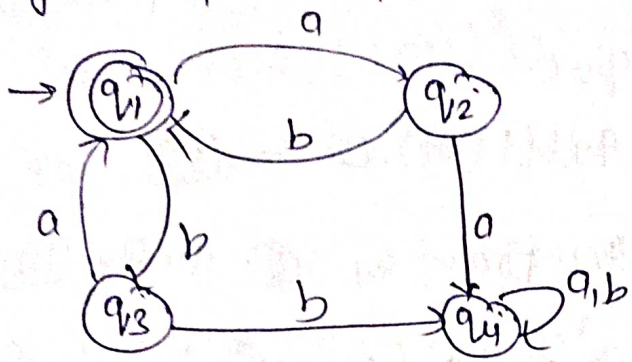
$$= Q + QP + (Q + RP)P^2$$

$$= Q + QP + QP^2 + QP^3 + \cdots$$

$$= Q(\epsilon + P + P^2 + P^3 + \cdots)$$

$$\boxed{R = QP^*}$$

Q. Find the regular expression for transition diagram given. (6)



Ans:

$q_1 = q_2b + q_3a + \epsilon$ — (1)

$q_2 = q_1a$ — (2)

$q_3 = q_1b$ — (3)

$q_4 = q_2a + q_3b + q_4a + q_4b$ — (4)

Eq. (2)

Keep the value of Eq. (3) & in to Eq. (2)

$q_1 = q_1ab + q_1aa + \epsilon$

$q_1 = q_1(ab+ba) + \epsilon$

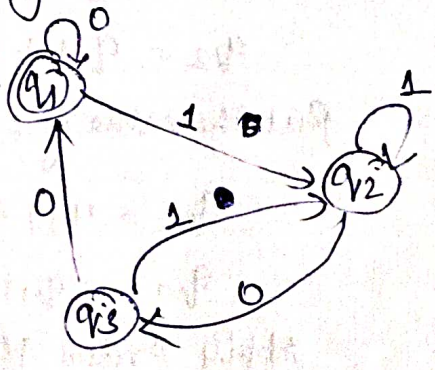
$q_1 = \underline{\epsilon} + \underline{q_1}(\underline{ab+ba})$

$R = Q + RP$

$q_1 = \epsilon(ab+ba)^* = (ab+ba)^*$

RE $\Rightarrow (ab+ba)^*$

Q. Construct a RE corresponding to the statediagrams.



Ans:

$q_1 = q_10 + q_30 + \epsilon$ — (1)

$q_2 = q_11 + q_21 + q_31$ — (2)

$q_3 = q_20$ — (3)

Put the value of q_3 in Eq. (2)

$q_2 = q_11 + q_21 + q_201$

$q_2 = q_11 + q_2(1+01)$

$q_2 = q_11(1+01)^*$ — (4)

Keep the value of q_3 from Eq. (4) to Eq. (3)

$$q_3 = q_20$$

$$q_3 = q_11(1+01)^*0 \quad \text{—(5)}$$

Keep the value of q_3 from Eq. (5) to Eq. (1)

$$q_1 = q_10 + q_11(1+01)^*00 + \epsilon$$

$$q_1 = q_1(0+1(1+01)^*00) + \epsilon$$

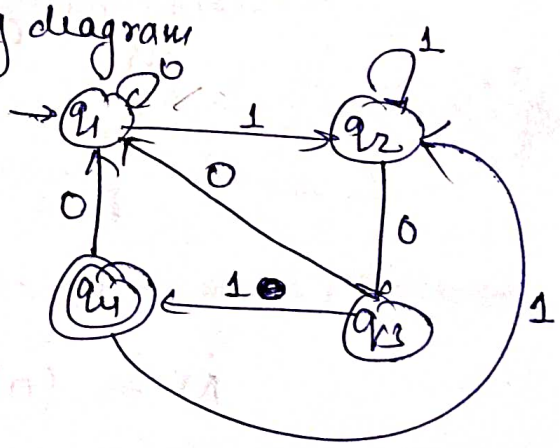
$$q_1 = \epsilon + q_1(0+1(1+01)^*00)$$

Apply the arden's theorum

$$R = Q + RP \Rightarrow QP^*$$

$$q_1 = (0+1(1+01)^*00)^*$$

Q. Find the regular expression corresponding diagram



$\Rightarrow$

$$q_1 = \epsilon + q_10 + q_30 + q_40 \quad \text{—(1)}$$

$$q_2 = q_11 + q_21 + q_41 \quad \text{—(2)}$$

$$q_3 = q_20 \quad \text{—(3)}$$

$$q_4 = q_31 \quad \text{—(4)}$$

Put the value of Eq. (4) into Eq. 2.

$$q_2 = q_11 + q_21 + q_311 \quad \text{—(5)}$$

Put the value of Eq. (3) in Eq. (5)

$$q_2 = q_11 + q_21 + q_2011$$

$$q_2 = q_11 + q_2(1+011)$$

Apply Arden theorum.

$$q_2 = q_11(1+011)^* \quad \text{—(6)}$$

Put the value of Eq. (6) into Eq. (3)

$$q_3 = q_11(1+011)^*0 \quad \text{—(7)}$$

Put the value of Eq. (7) into Eq. (4) (7)

$$q_4 = q_1 1(1+011)^* 0 \quad \text{—(8)}$$

Put the Eq. (8) and Eq. (7) value into Eq. (1)

$$q_1 = \epsilon + q_1 0 + q_1 1(1+011)^* 00 + q_1 1(1+011)^* 00$$

$$q_1 = \epsilon + q_1(0+1(1+011)^* 00 + 1(1+011)^* 00)$$

Apply Arden theorum

$$q_1 = (0+1(1+011)^* 00 + 1(1+011)^* 00)^* \quad \text{—(9)}$$

Put the value of Eq. (9) into Eq. (8)

$$q_4 = (0+1(1+011)^* 00 + 1(1+011)^* 00)^* 1(1+011)^*$$

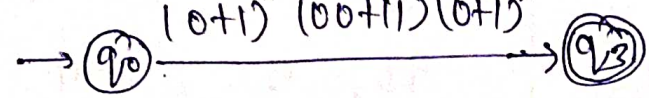
RE $\Rightarrow$ $(0+1(1+011)^* 00 + 1(1+011)^* 00)^* 1(1+011)^* 0$

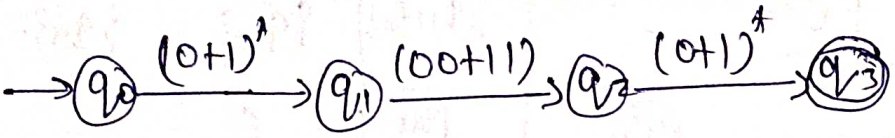
(Lecture-04)

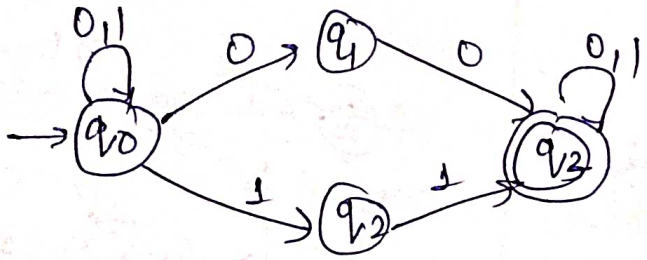
Q. Convert RE in FA.

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

Soln



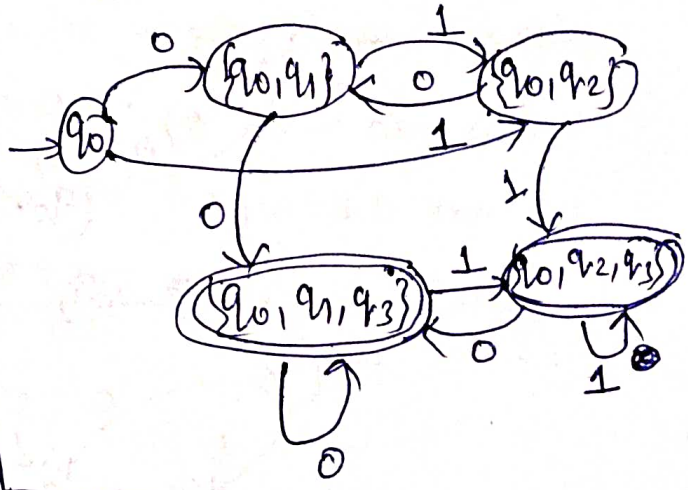




DFA TT

PS	N.S 0	1
→q0	{q0,q1}	{q0,q2}
{q0,q1}	{q0,q1,q3}	{q0,q2}
{q0,q2}	{q0,q1}	{q0,q2,q3}
{q0,q1,q3}	{q0,q1,q3}	{q0,q2,q3}
{q0,q2,q3}	{q0,q1,q3}	{q0,q2,q3}

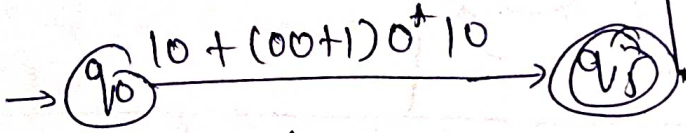
DFA

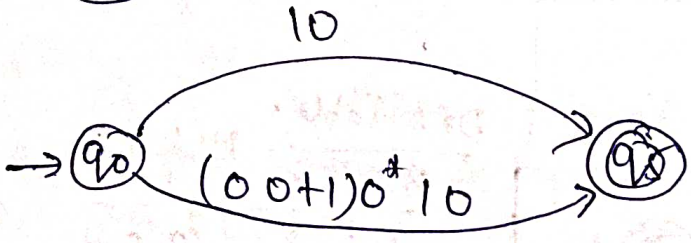


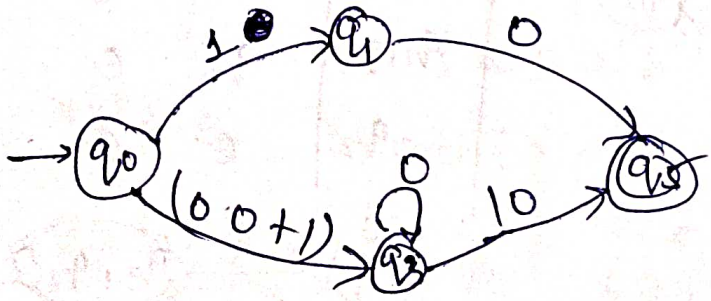
Q. Convert the RE in FA.

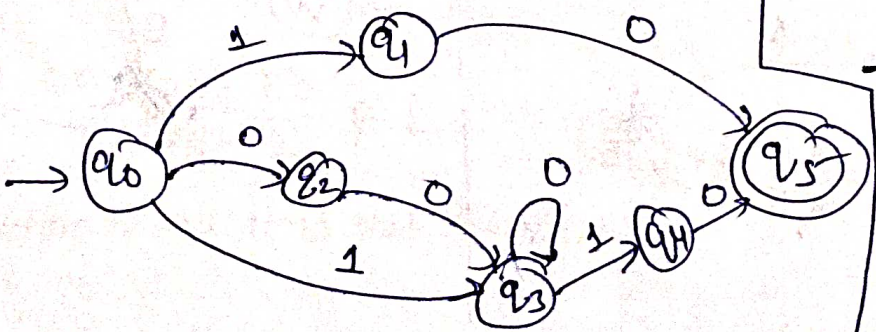
$10 + (00+1)0^*10$

Soln



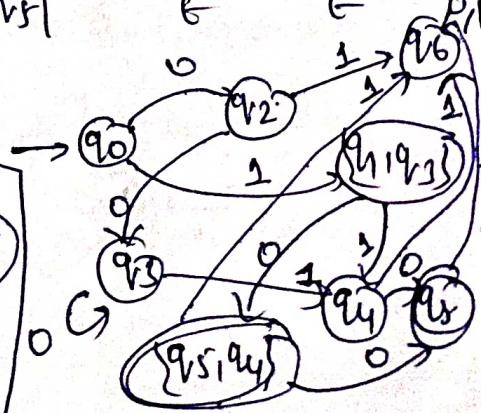






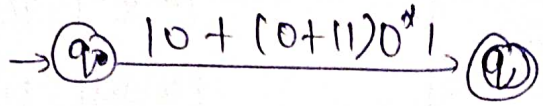
DFA Table

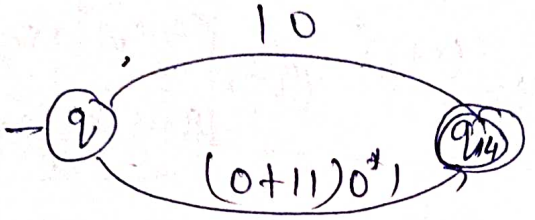
P.S.	N.S 0	1
→q0	q2	{q1,q3}
q2	q3	∈
{q1,q3}	{q5,q4}	{q4}
q3	q3	q4
{q5,q4}	q5	∈
q4	q5	∈
q5	∈	∈

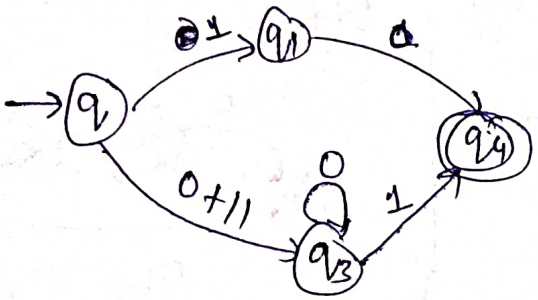


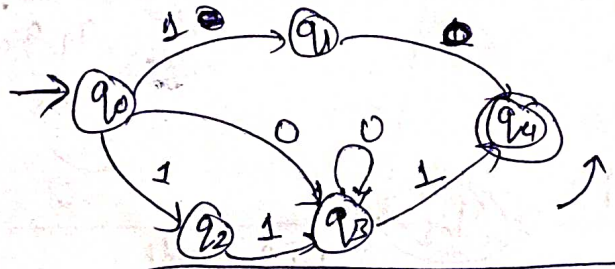
Q. Convert the Regular Exp. into FA.

$10 + (0+11)0^*1$



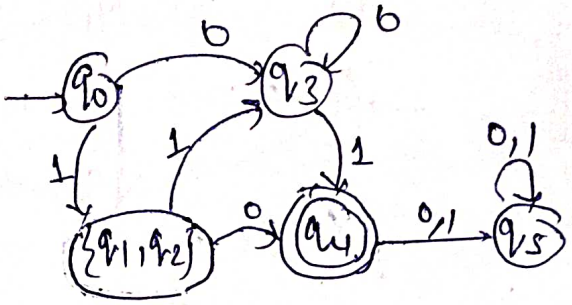






DFA Table: N-S

Q	0	1
→ q_0	$\{q_3\}$	$\{q_1, q_2\}$
q_3	q_3	q_4
$\{q_1, q_2\}$	$\{q_4\}$	$\{q_3\}$
(q4)	$\in$	$\in$

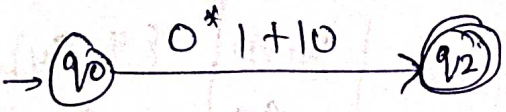


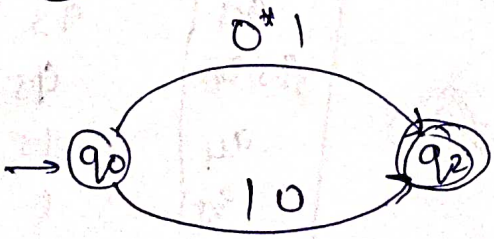
FA for RE

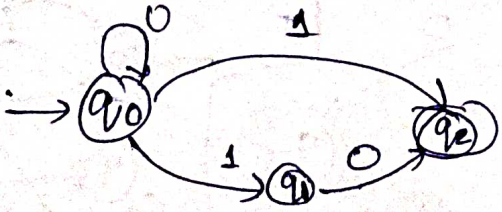
Q. Convert the RE in FA

$0^*1 + 10$

Ans

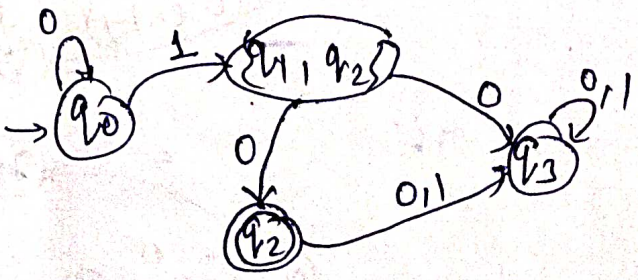






DFA Table: N-S

P.S	0	1
→ q_0	q_0	$\{q_1, q_2\}$
$\{q_1, q_2\}$	q_2	$\in$
q_2	$\in$	$\in$



(Lecture-05)

Pumping lemma for Regular expression

→ Pumping Lemma is used to prove that a language is NOT REGULAR
→ It can not be used to prove that a language is Regular

Language → Pumping Lemma Test —Pass→ undecidable (Reg / Not Reg)
↓ fail (Decidable)
Not Regular

Pumping Lemma:→ If L is an infinite language then there exists some positive integer 'n' (Pumping length) such that any string $w \in L$ has length greater than equal to 'n' i.e $|w| \geq n$ then w can be divided into three parts, $w = xyz$ satify following conditions.

i) for each $i \geq 0$, $xy^iz \in L$
ii) $|y| > 0$ and
iii) $|xy| \leq n$

Ex:→ $a^n b^{2n}$ $n \geq 0$ ∴no of b's is 2times of no of a's

Sol^n:→ Assume that Language is Regular
$w = \underset{x}{aa}\ \underset{y}{bbb}\ \underset{z}{b} \in L$ P = Pumping Length
$p = 2$ (Pumping length) -
$w = a^p b^{2p}$

i) $xy^iz \in L$ for each $i \geq 0$
2) $|y| > 0$
3) $|xy| \leq p$ $n = 2$ (P.L)

$\underset{x}{aa}\ \underset{y}{bb}\ \underset{z}{bb}$ $(bb)^i$ $i = 1$ $4 \leq p$ (Not satisfy)

$(bb)^i$ $i = 2$ (Pump up the value of y)

$\underset{x}{aa}\ \underset{y}{bbbb}\ \underset{z}{bb} \notin L$ (here no of b's is 3times of no of a's)

Contradiction→ so it is Not Regular.

Ex-2 Using pumping Lemma prove that the language $A = \{a^n b^n \mid n \geq 0\}$ is not Regular.

Proof:→ Assume that A is Regular

Pumping Length = P

$w = a^P b^P \Rightarrow w = aaaaaaa\,bbbbbbb$

x y z

P = 7

Case-1 The y is in the a part

aaaaaaa bbbbbbb

x y z

① $xy^iz \Rightarrow xy^2z$

aa aaaa aaaa a bbbbbbb

a's ≠ 7 b's

Case-2 The y is in the b part

aaaaaaa bbbbbbb

x y z

$xy^iz = xy^2z$

aaaaaaa bbbb bbbb bbb

7a ≠ 11 b

Case-3 The y is in the a' and b part

aaaaaaa bbbbbbb

x y z

$xy^iz \Rightarrow xy^2z$

aaaaa a abb aa bb bbbbb

↓

This does not follow the Pattern $a^n b^n$

So $xy^iz \notin L$

② $|y| > 0$

③ $|xy| \leq P \quad P = 7$

Case 1 $|xy| = 6$ (satisfy)

Case 2 $|xy| = 13$ (Not satisfy)

case 3 $|xy| = 9$ (Not satisfy)

Contradiction

So Language A is Not Regular.

Pumping Lemma for Reg. Language :-

Q Prove that $L = \{a^p \mid p \text{ is Prime}\}$ is not Regular.

Soln:- Let L is Regular language

Pumping Length N

Let $n = 3$

$w = aaa$

$w = \underset{x}{a}\underset{y}{a}\underset{z}{a}$

Prime No :->
* Divide by itself or only by 1
e.g. 2, 3, 5, 7, 9, 11...

$L = \{aa, aaa, aaaaa \cdots\}$

(i) $xy^iz \quad i \geq 0$

if $i = 0 \quad a \cdot a^0 \cdot a \Rightarrow aa \in L$

if $i = 1 \quad a \cdot a^1 \cdot a = \underset{y}{aa}a \in L$

if $i = 2 \quad a \cdot a^2 \cdot a = \underset{x}{a}\underset{y}{aa}\underset{z}{a} \notin L$

(ii) $|y| > 0 \quad 2 > 0$ (satisfy)

(iii) $|xy| \leq n \quad 3 \leq 3$ (satisfy)

so $xy^iz \notin L$ this is contradiction so language is not Regular.

(Lecture-06)

Regular Languages are Decidable

Regular languages are decidable: given any two regular languages A and B, an algorithm can determine whether A and B contain the same strings.

The decision algorithm exploits the fact that *set operations* can be performed on regular languages, based on transformations of finite automata. Because a regular language in any form (regular expression, DFA, and NFA) can be freely converted to any other form, these operations on automata are fully general.

Because these set operations yield a finite automaton as a result, the results of set operations on regular languages are also regular languages.

Let's say that A and B are finite automata recognizing regular languages A and B. (I.e., we use "A" to mean both a regular language and a finite automaton that recognizes language "A".) We can transform the automata A and/or B to construct an automaton that recognizes the languages

$A \cup B$, the union of A and B

A^c, the complement of A, meaning all strings (over some alphabet) that are *not* in A

$A \cap B$, the intersection of A and B

A - B (set difference), all strings that are in A but not in B

In addition to the possibility of performing set operations on finite automata, it is also possible to check any finite automaton to find out whether or not it recognizes a nonempty set of strings. Using this check, we can find out, for any regular language, whether or not that language is empty.

A decision procedure to check the equivalence of regular languages A and B is

$(A = B) \equiv ((A - B = \varnothing) \text{ and } (B - A = \varnothing))$

In other words, A is equivalent to B if and only if the set differences (A - B) and (B - A) are both empty.

Constructions for Set Operations

Here are constructions for set operations on regular languages.

Union (A ∪ B): The basic idea is to create new start and accepting states, connect the new start state to the start states for A and B using epsilon transitions, and connect the accepting states of A and B to the new accepting state using epsilon transitions. After these modifications are done, all of the original start and accepting states of A and B become non-start, non-accepting states. The resulting NFA recognizes the union of A and B.

Complement (A^C): Given a DFA recognizing language A, create an explicit "reject" state, which is a non-accepting state. In all states of A where there is no explicit transition on a particular input symbol, create an explicit transition to the reject state on that symbol. In the reject state, self-transitions on each possible input symbol lead back to the reject state. The result of these transformations is a DFA that recognizes the same language as the original DFA, but every state has a transition on every input symbol. By changing each accepting state to a non-accepting state, and changing each non-accepting state to an accepting state, we create a DFA that recognize the complement of A.

Intersection (A ∩ B): Intersection can be synthesized using the constructions for union and complement:

$$A \cap B = ((A^C) \cup (B^C))^C$$

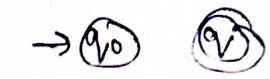
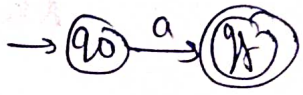
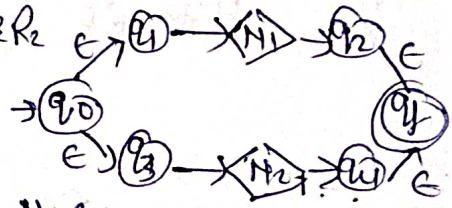
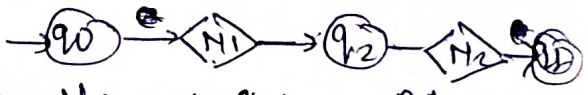
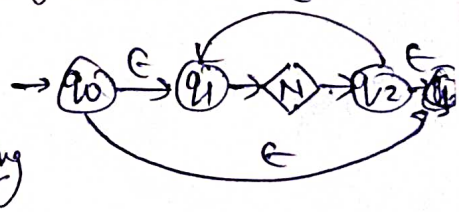
In other words, the intersection of A and B is the complement of the union of the complements of A and B.

Difference (A - B): Difference can be synthesized using the constructions for intersection and complement

$$A - B = A \cap B^C$$

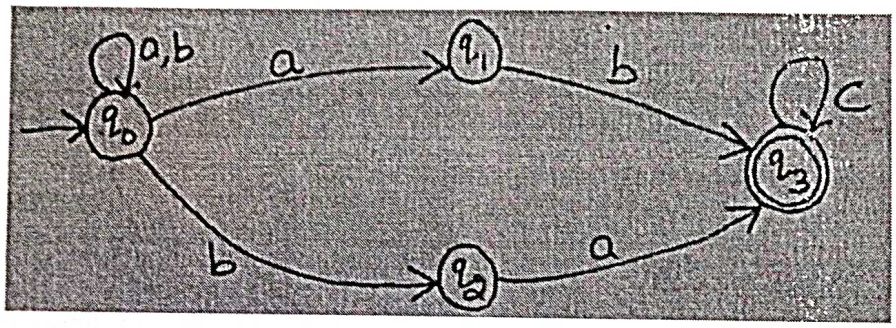
In other words, A - B is the set of all strings that are in A and also in the complement of B.

Comparative study of RE, Regular set & finite Automata

	Regular expression	Regular set	finite Automata
(1)	ϕ	$\{\}$	
(2)	ϵ	$\{\epsilon\}$	
(3)	Every a in Σ is a RE	$\{a\}$	
(4)	If r_1 & r_2 are regular expression then (r_1+r_2) is a RE	$R_1 \cup R_2$ {Where R_1 & R_2 are Regular set Corresponding r_1 & r_2 respectively	 N_1 & N_2 are FA accepting R_1 & R_2 respectively
(5)	If $r_1 . r_2$ is a RE	"	 Where N_1 & N_2 are FA accepting R_1 & R_2 resp.
(6)	r^* is a RE	R^* (Where R is the Regular set Corresponding to r)	 N ~~xxx~~ is FA accepting R.

UNIT II: (Important Questions)

1. State pumping lemma for regular languages. Use pumping lemma to prove that the language L, defined as follows, is not regular. L = {0m 1n | m and n are positive integers and m ≠ n}.
2. Explain the difference between Moore machine and Mealy machine. Describe with the help of an example how a Moore machine can be converted to a Mealy machine.
3. Write regular expression for set of all strings such that number of a's divisibleby 3 over Σ = {a,b}(UPTU 2018-19)
4. Let r1 and r2 be two regular expressions defined as follows :r1 = (00*1)*1 and r2 = 1 + 0(0 + 10)*11. Prove that r1 = r2. (UPTU 2012-13)
5. Prove that the language L = {0n | n is prime} is not regular.
6. Following Grammar generates language of Regular Expression :∈0B|1B|→ B∈0A/→A1B A→0*1 (0+1)* S Give leftmost and rightmost derivation of strings 00101.
7. Prove that following are not Regular Languages : (i) {on | n is perfect square}.
8. The set of strings of form 0i 1j such that the greatest common divisor of i and j is 1.
9. Design a Transducer (Mealy or Moore) Machine to compute multiplication of two n-bit binary numbers.
10. Write the statement of Pumping Lemma and using it prove that the language {$a^n b^n c^n$ | n>=1} is not regular. (UPTU 2013-14)
11. Findthe regularexpression corresponding tothe finite automatagiven bellow:(UPTU 2018-19)

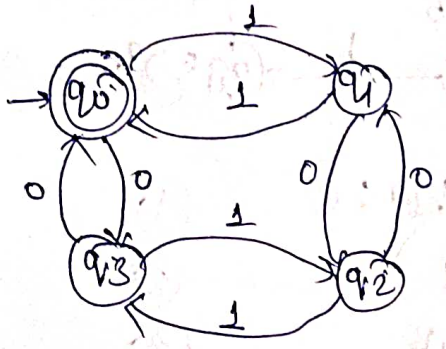


Kleen's Theorem:-

Kleen's Theorem has three Part:

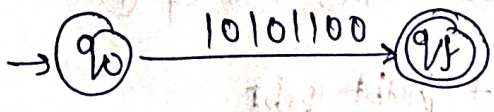
① if language can be accepted by FA (Finite Automata) then it can also be accepted by TG (Transition Graph)

Eg. Machin accept even no. of 0's & 1's for Input alphabate $\Sigma = \{0,1\}$



FA for abare Language

I/p string 10101100

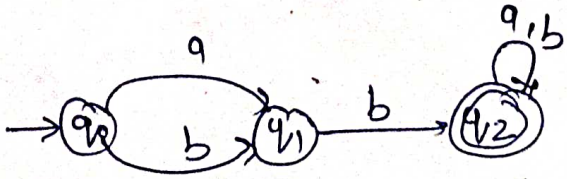


In above Eg. We draw a FA for given language and draw a TG (Transition Graph) which accept a set of string generated by Language.

② if language can be expressed by RE (Regular Expression) then it can be accepted by FA (Finite Automata)

Eg: Language that has b as second letter define over alphabate $\Sigma = \{a,b\}$

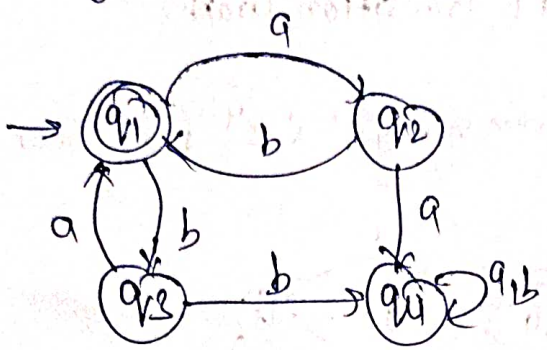
RE ⇒ $(a+b)\,b\,(a+b)^*$



In this Eg. we show we can draw a RE of the language and also draw FA of that RE.

③ If language can be accepted by TG (Transition Graph) then it can also be expressed by RE (Regular Expression) (Arden's theorem)

Eg. Transition graph of the language is



$q_1 = q_2b + q_3a + \epsilon$ — (1)

$q_2 = q_1a$ — (2)

$q_3 = q_1b$ — (3)

$q_4 = q_2a + q_3b + q_4a + q_4b$ — (4)

Keep the value of Eq. (2) & Eq. (3) into Eq. (1)

$q_1 = \epsilon + q_1ab + q_1ba$

$q_1 = \epsilon + q_1(ab + ba)$

$R = Q + R\underline{P}$

Apply Arden's theorem

$q_1 = \epsilon \cdot (ab+ba)^*$

$q_1 = (ab+ba)^*$

$RE = (ab+ba)^*$