Date
Page

Regular Expression

Regular Expressions are used for representing certain sets of strings in an algebraic fashion

1) Any terminal symbol i.e., Symbols $\in \epsilon$ including $\wedge$ and ϕ are regular Expressions

2) The union of two regular Expression is also a regular expression.

3) The concatenation of 2 Regular Exp. is also a regular expression

4) The Iteration (or closure) of a regular expression is also a regular Expression

5) The regular Exp. over Σ are precisely those obtained recursively by the application of the above rules once or several times.

Describe the following sets as Regular Exprs.

1) $\{0,1,2\}$ $R = 0+1+2$

2) $\{\wedge, ab\}$ $R = \wedge ab$

3) $\{abb, a, b, bba\}$ $R = abb + a + b + bba$

4) $\{\wedge, 0, 00, 000 \ldots\}$ $R = 0^*$ (Closure of zero)

5) {1, 11, 111, 1111 ... }

$R = 1^+$

Identities of Regular Exp.

1) $\phi + R = R$

2) $\phi R + R\phi = \phi$

3) $\epsilon R = R\epsilon = R$

4) $\epsilon^* = \epsilon$ and $\phi^* = \epsilon$

5) $R + R = R$ (Idempotent law)

6) $R^* R^* = R^*$

7) $RR^* = R^* R = R^+$

8) $(R^*)^* = R^*$

9) $\epsilon + RR^* = \epsilon + R^*R = R^*$

10) $(PQ)^* P = P(QP)^*$

11) $(P+Q)^* = (P^* Q^*) = (P^* + Q^*)^*$

12) $(P+Q)R = PR + QR$ and

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

Closure under union (+)
Closure under kleene's (*)
Closure under intersection (.)
" " " difference (-)
" " " complement ($\Sigma^* - L$)

Note
$RR^* = R^+$
$\therefore R^+ + \epsilon = R^*$

Date
Page

Arden's Theorem

If P and Q are two Regular Expressions over Σ, and if P does not contain Σ, then the following equation in R given by $R = Q + RP$ has a unique solution i.e., $R = QP^*$

$$R = Q + RP \quad \text{— (1)}$$

let

$= Q + QP^*P$ → $R = QP^*$

$= Q(\Sigma + P^*P)$ → Take Q as common

$= QP^*$ → using Identity $\Sigma + P^*P = P^*$

Proved that $R = QP^*$ is a soln to this eqn (1)

To Prove it is unique soln

$R = Q + RP$

let $R = Q + RP$

$\therefore R = Q + [Q + RP]P$

$= Q + QP + RP^2$ → again Replace R.

$= Q + QP + [Q + RP]P^2$

$= Q + QP + QP^2 + RP^3$

$= Q + QP + QP^2 + \cdots QP^n + RP^{n+1}$ doing this n no. of time

Now Replace R with (QP^*)

$= Q + QP + QP^2 - - + QP^n + QP^*P^{n+1}$

Take Q common.

$R = Q[\epsilon + P + P^2 + \cdots + P^n + P^* P^{n+1}]$

This whole term represents Closure of P.

∴

$R = QP^*$

Hence it is proved that.

$R = QP^*$ is the unique Sol^n of $R = Q + RP$

Q. Example proof using Identities of Reg. Exp.

PT $[1 + 00^*1) + (1 + 00^*1)(0 + 10^*1)^*(0 + 10^*1)$ is equal to $0^*1(0 + 10^*1)^*$

L.H.S $= (1 + 00^*1) + (1 + 00^*1)(0 + 10^*1)^*(0 + 10^*1)$

Take common $(1 + 00^*1)$

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

use Identity $\epsilon + RR^* = R^*$

$= (1 + 00^*1)[(0 + 10^*1)^*]$

use Iden. $\epsilon \cdot R = R$

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

Take common ϵ and 00^*

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

use Iden. $\epsilon + RR^* = R^*$

$= 0^*1(0 + 10^*1)^* =$ R.H.S.

= proved

Note:-

1. The intersection of two regular set is regular
2. The union of two regular set is regular.
3. The complement of Regular set is regular.

Designing Regular Expression

Design Reg. Exp. for the follow lang. over {a,b}

1) lang. accepting strings of length exactly 2

$L_1 = \{aa, ab, ba, bb\}$

$R = aa + ab + ba + bb$

$= a(a+b) + b(a+b)$

$= (a+b)(a+b)$

2) lang. accepting strings of length at least 2.

$L_2 = \{aa, ab, ba, bb, aaa, aab, bba, bbb, aba, bab \ldots\}$

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

(TWO or more than 2)

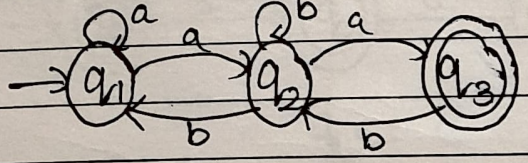
3) lang. accepting strings of length at most 2.

$L_3 = \{\epsilon, a, b, aa, bb, ab, ba\}$

$R = \epsilon + a + b + aa + bb + ab + ba$

$= (\epsilon + a + b)(\epsilon + a + b)$

Find Reg. Expression for the given NFA.



$q_3 = q_2 a \rightarrow (1)$

$q_2 = q_2 b + q_1 a + q_3 b \rightarrow (2)$

$q_1 = \epsilon + q_1 a + q_2 b \rightarrow (3)$

Now Simplify States.

(1) $\rightarrow q_3 = q_2 a$

$= (q_1 a + q_2 b + q_3 b) a$

$= (q_1 aa + q_2 ba + q_3 ba) \rightarrow 4$

Date
Page

(2) → $q_2 = q_1a + q_2a + q_3b$ (Putting value of q_3 from eqn (1))

$= q_1a + q_2b + (q_2a)b$

$= q_1a + q_2b + q_2ab.$

$q_2 = q_1a + q_2(b+ab)$

R Q R P

By ardens Theorem.

$q_2 = (q_1a)(b+ab)^*$ — (5)

(3) → $q_1 = \epsilon + q_1a + q_2b$

putting value of q_2 from (5)

$q_1 = \epsilon + q_1a + ((q_1a)(b+ab)^*)b$

R Q R P

$q_1 = \epsilon + q_1(a + a(b+ab)^*)b$

By ardens Theorem

$q_1 = \epsilon(a + a(b+ab)^*)b)^*$

$\because \epsilon \cdot R = R$

$q_1 = (a + a(b+ab)^*)b)^*$ — (6)

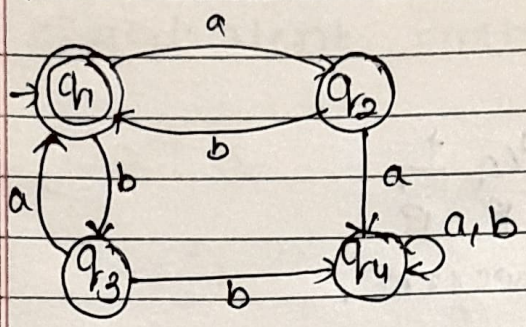
Final state

$q_3 = q_2a$

$= q_1a(b+ab)^*a$ Put value q_2 from eqn (5)

$= (a+a(b+ab)^*)b)^*a(b+ab)^*a$ Put value q_1 from eqn (6)

Q Find the R.E for the following DFA.



$q_1 = \epsilon + q_2 b + q_3 a$ — (I)

$q_2 = q_1 a$ — (II)

$q_3 = q_1 b$ — (III)

$q_4 = q_2 a + q_3 b + q_4 a + q_4 b$ — (IV)

(I) → $q_1 = \epsilon + q_2 b + q_3 a$

from eqn (2) and (3)

$q_1 = \epsilon + q_1 aba + q_1 ba$

Taking common q_1

$$\underbrace{q_1}_{R} = \underbrace{\epsilon}_{Q} + \underbrace{q_1}_{R}\underbrace{(aba + ba)}_{P}$$

So by arden's theorem

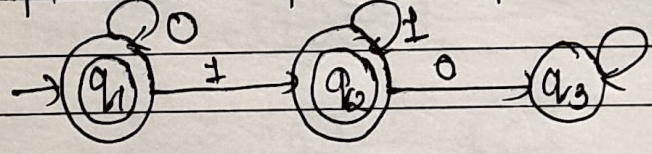
$q_1 = \epsilon (aba + ba)^*$

also $\epsilon \cdot R = R$

$\therefore q_1 = (aba + ba)^*$

$\because q_1$ is the final state hence Req. reg. Exp. for given DFA

Q. Find R.E for the following DFA having Multiple final states



$q_1 = \epsilon + q_1 0$ — (1)

$q_2 = q_1 1 + q_2 1$ — (2)

$q_3 = q_2 0 + q_3 0 + q_3 1$ — (3)

(1) →

$$\underbrace{q_1}_{R} = \underbrace{\epsilon}_{Q} + \underbrace{q_1}_{R}\underbrace{0}_{P}$$ — (1)

$\therefore q_1 = \epsilon \cdot (0)^*$

as $\epsilon \cdot R = R$ $\therefore q_1 = 0^*$ — (4)

$q_{r2} = q_{r1} + q_{r2}1$

from eqn 4

$\underbrace{q_{r2}}_{R} = \underbrace{0^*1}_{Q} + \underbrace{q_{r2}}_{R}\underbrace{1}_{P}$

∴ By Arden's Theorem.

$q_{r2} = 0^*1(1)^*$

∵ we have two final states.

∴ R.Exp = union of both final states.

$= q_{r1} + q_{r2}$

$= 0^* + 0^*1(1)^*$

$= 0^*(\epsilon + 11^*)$

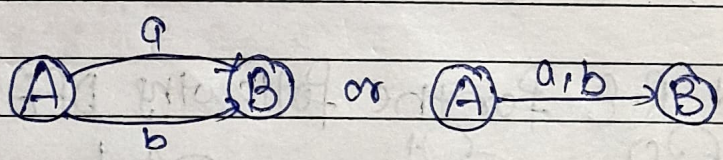
also $\epsilon + RR^* = R^*$

∴ $R.E. = 0^*1^*$

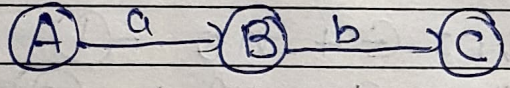
Req. R.Exp for DFA given

Conversion of RE to F.A

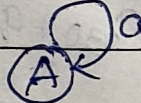
(1) (a+b)



(2) (a.b)



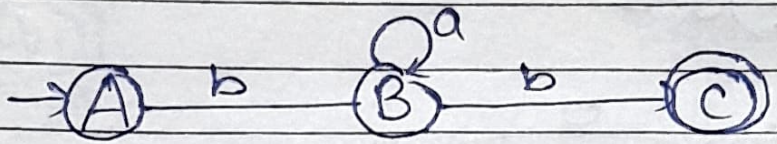
(3) a^*



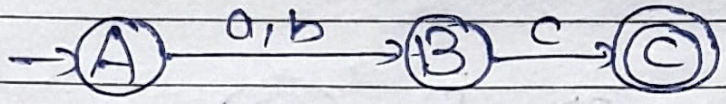
Date
Page

Q Convert the following Regular Expressions to their equivalent finite Automata.

1) ba^*b

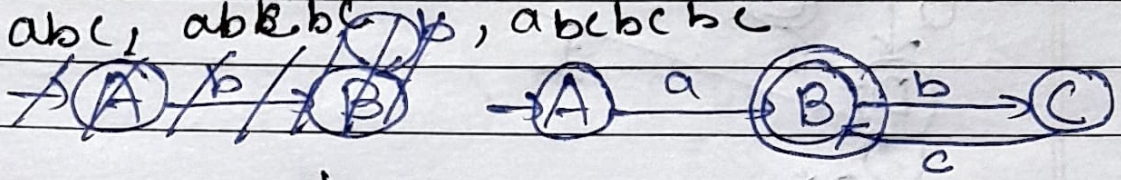


2) $(a+b)c$



3) $a(bc)^*$

$L = a, abc, abbc, abbbc, \ldots, abcbcbc$

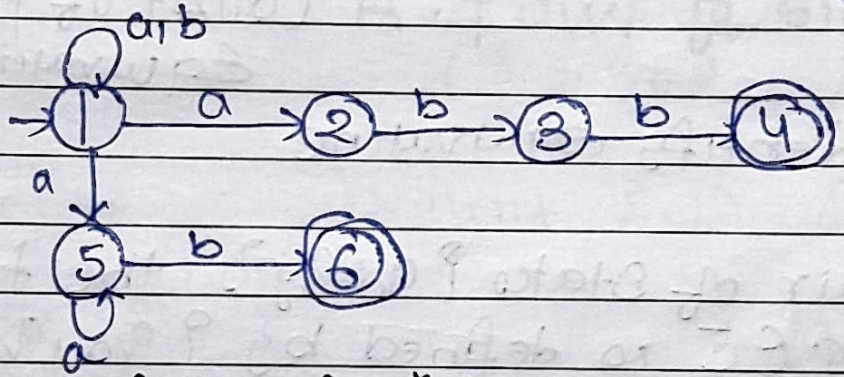


4) $(a|b)^*(abb|a^+b)$

a^+ = at least 1 a.

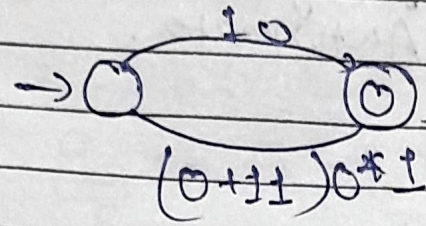
$a^+ = \{a, aa, aaa \ldots\}$

$a^* = \{\epsilon, a, aa \ldots\}$

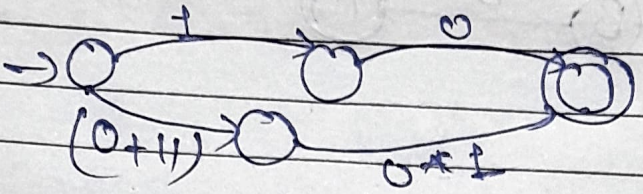


Q5. $10+(0+11)0^*1$

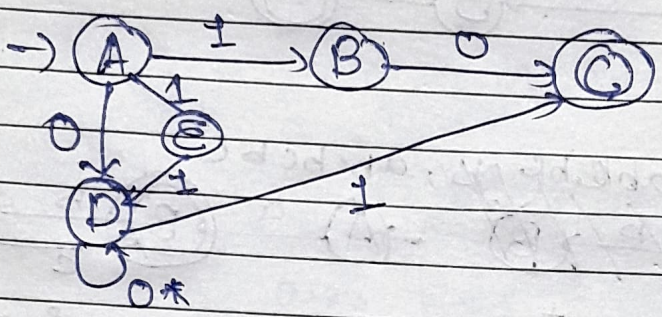
S1



S2



S3



Equivalence of two F.A (Gist of prev. Equivalence Qus)

Steps to identify equivalence

1) for any pair of states $[q_i q_j]$ the transition for Input $a \in \Sigma$ is defined by $[q_a, q_b]$ where

$$\delta[q_i, a] = q_a \text{ and } \delta[q_j, a] = q_b$$

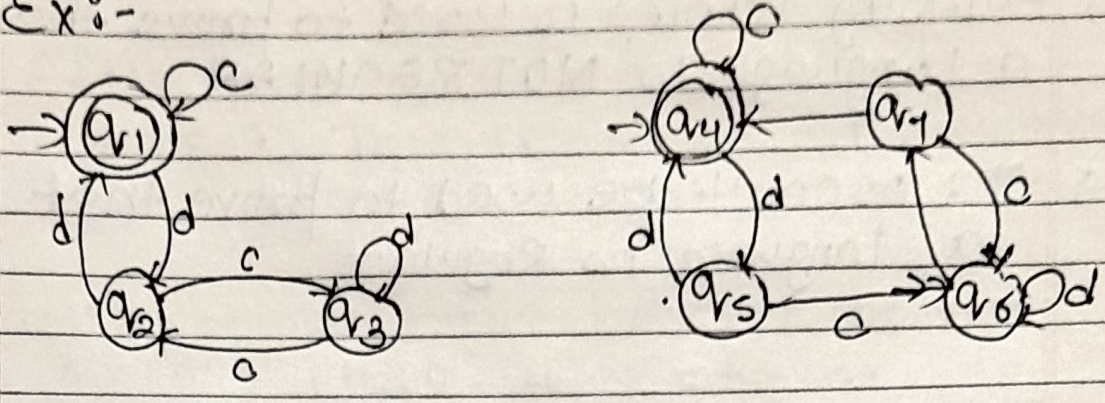
The two automata are not equivalent if for a pair $[q_a, q_b]$ one is Intermediate State and other is final State.

2) If initial State is final state in one of automaton then second also Initial state must be final State

for them to be equivalent.

Ex:-

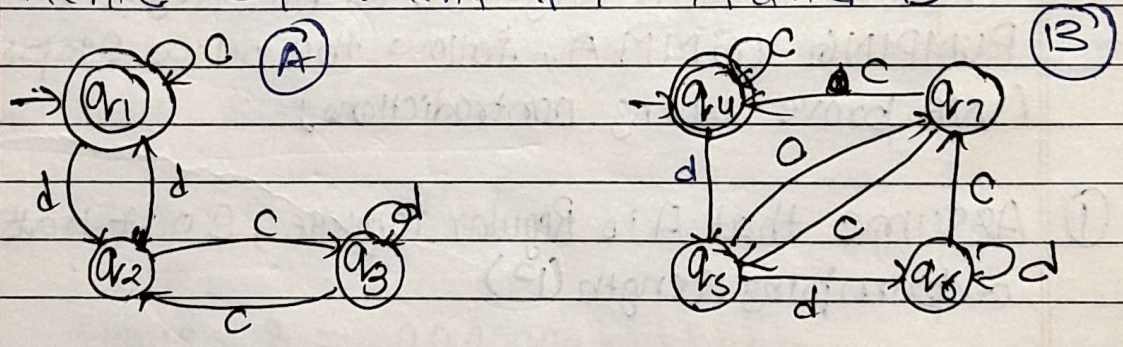
9.



States	c	d	
$[q_1 q_4]$	(q_1, q_4) f.s f.s	$(q_2 q_5)$ I.S I.S	✓
$[q_2 q_5]$	$(q_3 q_6)$ I.S I.S	$(q_1 q_4)$ f.s f.s	✓
$[q_3 q_6]$	$(q_2 q_7)$ I.S I.S	$(q_3 q_0)$ I.S I.S	✓
$[q_2 q_7]$	$(q_3 q_0)$ I.S I.S	$(q_1 q_4)$ F.S F.S	✓

Hence equivalent f.A A and B

9.



States	c	d	
$[q_1 q_4]$	(q_1, q_4) f.s f.s	$(q_2 q_5)$ I.S I.S	✓
$[q_2 q_5]$	$(q_3 q_7)$ I.S I.S ✓	(q_1, q_0) f.s I.S	✗

The given A and B automata are Not Equivalent

Date
Page

Pumping Lemma (for Regular Languages)

→ Pumping lemma is used to prove that a Language is NOT REGULAR

→ It can't be used to prove that a language is Regular.

If A is a Regular language, then A has a pumping length 'P' such that any string 'S' where $|S| \geq P$ may be divided into 3 parts $S = xyz$ such that the following conditions must be true:

1) $xy^iz \in A$ for every $i \geq 0$
2) $|y| > 0$
3) $|xy| \leq P$

To prove that a language is not Regular using PUMPING LEMMA, follow the below steps: (We prove using contradictions)

① Assume that A is Regular language, so it has a pumping length (P)

② Now find a string 'S' in A such that it satisfies $|S| \geq P$

③ Divide S into XYZ such that i) $|y| > 0$
ii) $|XY| \leq P$

Date
Page

④ Now Show that
$$xy^iz \notin A \text{ for Some } i$$

⑤ After proving $xy^iz \in A$
Write :- This contradicts our assumption
hence A is not Regular language

Q. Using pumping lemma prove that the language $A = \{0^n b^n \mid n \geq 0\}$ is Not Regular.

Why not Theory :- In order to achieve this Language we need to keep track of count of how many A's we will have then only we can repeat the same no. of B's But we know that finite State M/C can't keep count of anything.
∴ language can't be designed using F.S.M

Why not proof :-

① Assume that A is Regular, so it has a pumping length say (P).

② String $S = a^P b^P$
Let P = 7
Hence S = aaaaaaabbbbbbb

③ Divide S into xyz such that i) $|y| > 0$
ii) $|xy| \leq P$ i.e., 7

Hence S = $\underbrace{aaa}_{x}\underbrace{aa}_{y}\underbrace{bbbbbbb}_{z}$

Date
Page

④ $S = \underbrace{aaaaaa}_{x}\underbrace{a}_{y}\underbrace{bbbbbbb}_{z}$

$xy^iz = aaaaa\ aa\ bbbbbbb$

Let $i = 2$

$\therefore xy^iz = aaaaa\ aaa\ bbbbbbb$

Condition of $a^n b^n = |a| = |b|$

Now

$|a| = 9$ and $|b| = 7$

Hence $|a| \neq |b|$ if xy^2z

$\therefore xy^2z \notin A$ for i as (2)

⑤ This contradicts our assumption

Hence A is not a Regular language.

Q. Using pumping lemma prove that the language $A = \{yy \mid y \in \{0,1\}^*\}$ is Not Regular.

Theory:- The first part of string should be same as 2nd part of string. In order to repeat the first part exactly in the 2nd part F.S.M needs to remember what was the first part ∵ F.S.M can't have large Memory ∴ It can't be regular language.

Proof:- ① Assume that A is a regular language, so it has a pumping length say ($p = 7$)

② String $S = 0^p 1 0^p 1$

$\therefore S = 0000000\,1\,0000000\,1$

Date
Page

③ Divide S into XYZ such that i) $|y| > 0$
ii) $|xy| \leq p$

$\therefore S = \underbrace{0000000}_{x}\underbrace{0}_{y}\underbrace{10000001}_{z}$

~~or~~

~~$S = \underbrace{0000000}_{z}\underbrace{0}_{y}\underbrace{10000001}_{x}$~~

④ $\because S = \underbrace{00000}_{x}\underbrace{00}_{y}\underbrace{100000001}_{z}$

$xy^i z = 00000\ 0000\ 100000001$ if $i = 2$

1st 2nd

Condition of Lang → $|0| = |0|$
in 2 Halfs. → $|1| = |1|$

Now, if $i = 2$ 1st half $|0| = 9$ and $|1| = 1$
2nd half $|0| = 7$ and $|1| = 1$

$\therefore xy^2z \notin A$ if $i = 2$

⑤ This contradicts our assumption Hence A is not a Regular language

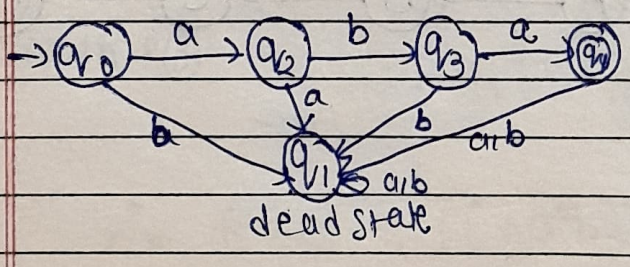
Some missed topics.

Kleen's Theorem

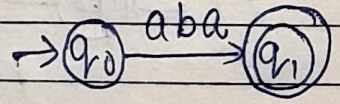
① If language can be accepted by F.A (Finite Automata) then it can also be accepted by TG (Transition Graph)

Ex:- Let S = aba

NFA/DFA



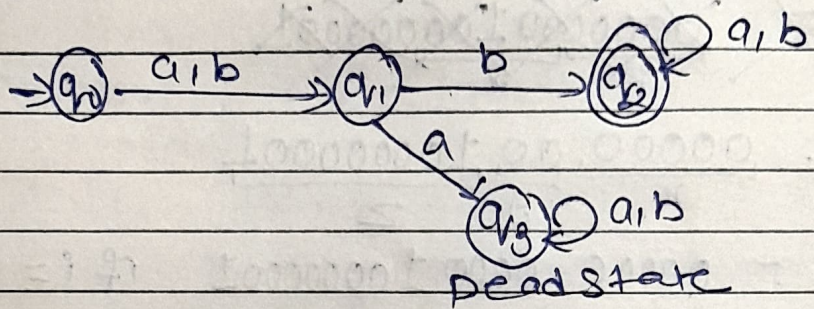
TG



Date
Page

(2) If language can be expressed by R.E (Regular Expression) then it can be accepted by FA (Finite Automata)

Ex:- R.E = (a+b) b (a+b)*



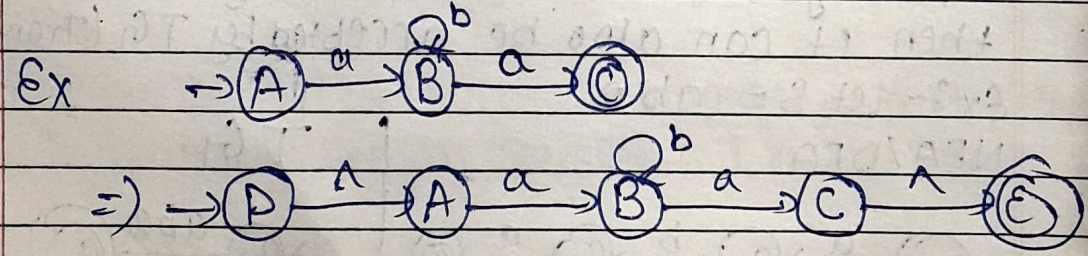
(3) If language can be accepted by TG then it can also be expressed by RE

Some Rules to keep in mind which is similar to conversion of R.E to F.A.

(1) ~~If~~ Introduce a new initial state and final state and connect it with old initial state and final state respectively by the transition labelled as (∧)

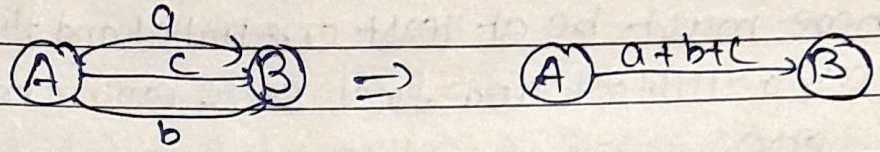
for making the old initial state as Non-Initial State
Old final state as Non-final state

Ex

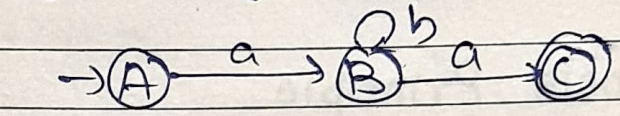


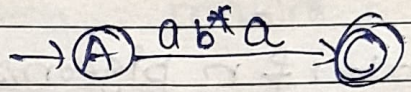
=>

(2) If a state has an option of (two or more than two) incoming transition edges, then replace all these transition edges with a single (union) transition denoted by (+) symbol

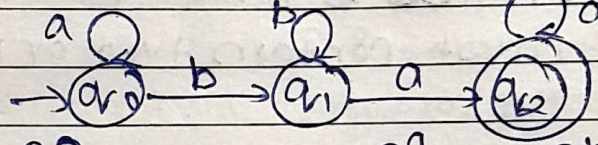


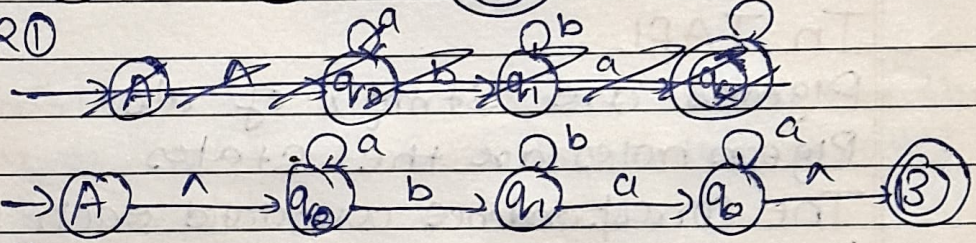
(3) If 3 states in a TG, are connected in sequence then eliminate the middle state and connect the first state with the third by a single transition.

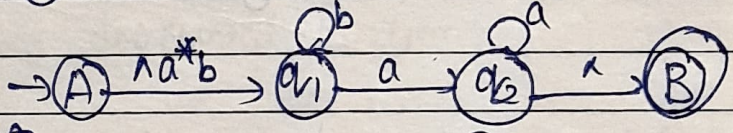
Ex:- 

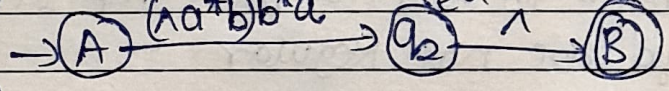
=> 

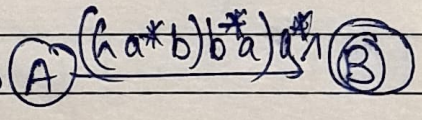
Proof.

Ex:- 

Apply R(1) 

Apply R(3) 

Apply R(3) 

again R(3) 

$\therefore R.E = ((\wedge a^*b)b^*a)a^*\wedge$

Transition Graph.

It is a method to define a language

Prob.

1. No. of states should be finite
2. There must be at least one initial and final states. also initial and final state can be more than one.
3. No dead state
4. Multiple transition are allowed.
5. Can read more than one character at a time
6. It allows null transition.

Pigeonhole Principle

It states that if n pigeons fly into m pigeon holes & $n > m$ then atleast one hole must contain two or more pigeons

In TAFL

Pigeons are Strings of a's

Pigeon holes are the states

The correspondence associated each string with the state to which A goes when the string is i/p

This principle is used to prove that certain infinite languages are not Regular

Ex:- $L = \{a, aa, aab\}$ → no state is repeated

or

$L = \{aabb, bbaa, abbabb, abbbabbabbb\}$

→ a state has been repeated

If. String w

has length $|w| \geq$ no. of states.

then the transition of string w are more than the states of F.A

So a state (q) must be repeated

in the transition of w

Decidability

- Decidability is making decision about following

1) IS given F.A/R.E accept any string or not?
2) IS given F.A/R.E represents finite or infinite long?
3) Whether two languages are equivalent or not?

- The problems that solve in finite steps and their answer is Yes or NO, such problems are called decidable problems.

3) already discussed earlier.

1) Decide whether given F.A/R.E accept any string or not?

To find out that given FA/RE accept any string or
→ we will perform some steps on F.A either given or constructed from R.E

Steps :-

1. Mark the initial state
2. Mark the states that are connected with initial states.
3. Remove the edges that connect initial state with the other states.
4. Remove the edges of next marked state and if next marked state edges connected with another state then also marked that state
5. Repeat the above process until final state is marked
6. a) If we are unable to reach final state, it means our F.A does not accept any string.

   b) If we reach to final state, it means our F.A accept any string.

Upon this condition (a)

write :- After performing the desired steps, our final Automata will not reach final state.
Hence this concludes that the given F.A will not accept any string.

Upon this condition (b)

write :- After performing the desired steps our final Automata will reach final state
Hence it concludes that the given F.A will accept any string.

Date
Page

2) Decide whether given R.E / FA represents finite or infinit language?

Case 1: we have given R.E, We have to check is given R.E represents finit or infinit language.

- If there is no kleen star (*) in given R.E, then it must be represent finite language.

  Ex:- R.E = (a+b) b (a+b)

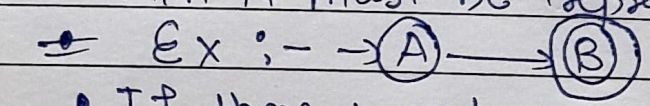
- If there is kleen star (*) in given R.E, then it may represent finite or infinite language.
  - If we have kleen star only on Null, we can say R.E must Represent finite lang.

    Ex:- R.E = (a+^*) a (b+^*) b
  - If we have kleen star other than Null we can say R.E must represent infinite langue.

    Ex:- R.E = (a+b)* a (b+a) b*

Case 2: we have given FA, We have to check is given FA represents finite or infinite langue.

- If there is no loop or circuit loop in given FA, then it must be represent finite language.

  Ex:- 

- If there is loop or circuit loop in given FA, then it must be represent infinite langue.

  Ex:-