

Regular Expression

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

- 1) Any terminal symbol i.e., symbols $\in E$ including λ and \emptyset 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 E are precisely those obtained recursively by the application of the above rules once or several times.

Describe the following sets as Regular Express.

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

2) { $a^k b^l$, $a^m b^n$ } $R = a^k b^l + a^m b^n$

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

4) { $1, 0, 00, 000\dots$ } $R = 0^*$ (closure of zero)

5) $\{1, 11, 111, 1111, \dots\}$

$$R = 1^+$$

Identities of Regular Exp.

- 1) $\emptyset + R = R$ Closure under union (+)
- 2) $\emptyset R + R \emptyset = \emptyset$ Closure under kleenstar (*)
- 3) $\epsilon R = R \epsilon = R$ Closure under intersection (.)
1 1 1 1 1 difference (-)
1 1 1 1 complement ($\epsilon^* - \cup$)
- 4) $\epsilon^* = \epsilon$ and $\emptyset^* = \epsilon$
- 5) $R + R = R$ (Idempotent law)
- 6) $R^* R^* = R^*$
- 7) $RR^* = R^* R = R^+$
- 8) $(R^*)^* = R^*$
- 9) $\epsilon + RR^* = \epsilon + R^* R = R^*$ | NOTE
 $RR^* = R^+$
 $\therefore R^+ + \epsilon = R^*$
- 10) $(PQ)^* P = P(QP)^*$
- 11) $(P+Q)^* = (P^* Q^*) = (P^* + Q^*)^*$
- R) $(P+Q)R = PR + QR$ and
 $R(P+Q) = RP + RQ$

Arden's Theorem

If P and Q are two regular expressions over Σ , and if P does not contain ϵ , 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

$$= Q + QP^*P \rightarrow R = QP^*$$

$$= Q(\epsilon + P^*P) \rightarrow \text{Take } Q \text{ as common}$$

$$= QP^* \quad \rightarrow \text{using Identity} \\ \underline{\qquad\qquad\qquad} \quad \epsilon + P^*P = P^*$$

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

To Prove it is unique solⁿ

$$R = Q + RP$$

$$\text{let } R = Q + RP$$

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

$$= Q + QP + RP^2 \rightarrow \text{again Replace } R,$$

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

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

$$\begin{array}{ccccccc} | & & | & & | & & | \\ & & & & & & \\ & & & & & & \end{array}$$

$$= Q + QP + QP^2 + \dots + QP^n + RP^{n+1} \quad \text{doing this } n \text{ no. of times}$$

Now Replace

R with $(QP)^*$

$$= Q + QP + QP^2 + \dots + QP^n + QP^{n+1}$$

Take Q common.

$$R = Q \underbrace{[C + P + P^2 + \dots + P^n + P^n P^{n+1}]}_{\text{Closure of } P}$$

This whole term represents
closure of P.

∴

$$R = Q P^*$$

Hence \uparrow proved that.

$R = Q P^*$ is the unique Solⁿ of $R = Q + RP$

Q. Example proof using Identities of Reg. Exp.

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

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

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

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

use Identity $C + R R^* = R^*$

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

use Iden. $C \cdot R = R$

$$= (C \cdot I + 00^* I) (0 + 10^* I)^*$$

② Take common C and 00^*

$$= (C + 00^*) I (0 + 10^* I)^*$$

use Iden. $C + R R^* = R^*$

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

===== 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

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

$$\begin{aligned} R &= a\alpha + b\beta + \beta\alpha + \beta\beta \\ &= a(a+b) + b(a+b) \\ &= (a+b)(a+b) \end{aligned}$$

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

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

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

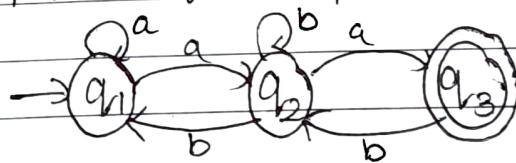
TWO or more than 2.

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

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

$$\begin{aligned} R &= \epsilon + a + b + aa + bb + ab + ba \\ &= (\epsilon + a + b)^* (\epsilon + a + b) \end{aligned}$$

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.

$$\begin{aligned} ① \rightarrow q_3 &= q_2 a \\ &= (q_1 a + q_2 b + q_3 b)a \\ &= (q_1 a a + q_2 b a + q_3 b a) \rightarrow ④ \end{aligned}$$

$$\textcircled{2} \rightarrow q_0 = q_1 a + q_2 b + q_3 ab \quad (\text{Putting value of } q_3 \text{ from eqn ①})$$

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

$$= q_1 a + q_2 b + q_1 a b.$$

$$\frac{q_2}{R} = \frac{q_1 a}{Q} + \frac{q_2 (b+ab)}{R - P}$$

By Arden's Theorem,

$$q_2 = (q_1 a)(b+ab)^* \quad \text{--- ⑤}$$

$$\textcircled{3} \rightarrow q_1 = E + q_1 a + q_2 b$$

Putting value of q_2 from ⑤

$$q_1 = E + q_1 a + ((q_1 a)(b+ab)^*) b$$

$$\frac{q_1}{a} = E + \frac{q_1 a}{R} \left(a + a(b+ab)^* \right) b$$

By Arden's Theorem

$$q_1 = E \{ a + a(b+ab)^* b \}^*$$

$\because E \cdot R = R$

$$q_1 = (a + a(b+ab)^* b)^* \quad \text{--- ⑥}$$

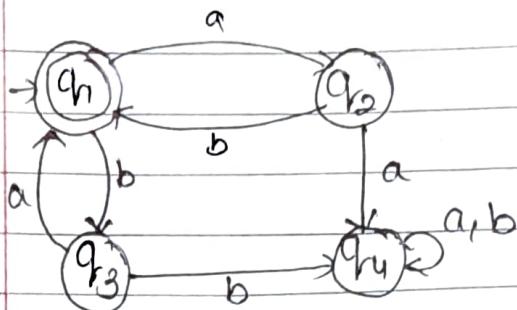
Final state

$$q_3 = q_2 a$$

$$= q_1 a (b+ab)^* a \quad \text{put value } q_2 \text{ from eqn ⑤}$$

$$= (a + a(b+ab)^* b)^* a (b+ab)^* a \quad \text{, } q_1 \text{ from eqn ⑥}$$

Q. Find the R.E. for the following DFA.



$$q_1 = \epsilon + q_2 b + q_3 a \quad \text{--- (1)}$$

$$q_2 = q_1 a \quad \text{--- (2)}$$

$$q_3 = q_1 b \quad \text{--- (3)}$$

$$q_4 = q_2 a + q_3 b + q_4 a + q_4 b \quad \text{--- (4)}$$

$$\text{From eqn (2) and (3)}$$

$$(1) \rightarrow q_1 = \epsilon + q_2 b + q_3 a$$

$$q_1 = \epsilon + q_1 abt q_1 ba$$

Taking common q_1

$$\frac{q_1}{R} = \frac{\epsilon}{a} + \frac{q_1}{R} (abtba)$$

So by Arden's theorem

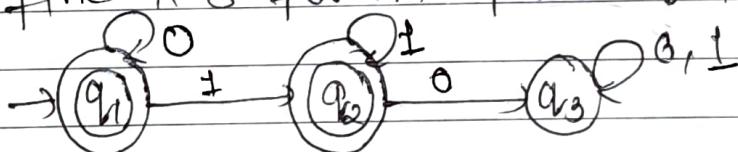
$$q_1 = \epsilon (abtba)^*$$

$$\text{also } E \cdot R = R$$

$$\therefore q_1 = (abtba)^*$$

$\therefore q_1$ is the final state hence Req. deg. Exp. for given DFA

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



$$q_1 = \epsilon + q_1 0 \quad \text{--- (1)}$$

$$q_2 = q_1 1 + q_2 1 \quad \text{--- (2)}$$

$$q_3 = q_2 0 + q_3 0 + q_3 1 \quad \text{--- (3)}$$

$$\frac{q_1}{R} = \frac{\epsilon}{a} + \frac{q_1}{R} 0 \quad \text{--- (1)}$$

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

$$\text{as } E \cdot R = R \therefore q_1 = 0^* \quad \text{--- (4)}$$

$$q_2 = q_{1,1} + q_{2,1}$$

From eqn 4

$$\frac{q_2}{R} = \frac{0^* \perp}{a} + \frac{q_2 \perp}{R \perp}$$

\therefore By Arden's Theorem,

$$q_2 = 0^* I (I)^*$$

\therefore we have two final states.

$\therefore R \cdot E \cup p = \text{union of both final states}$

$$= q_1 + q_2$$

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

$$= 0^* (C + II^*)$$

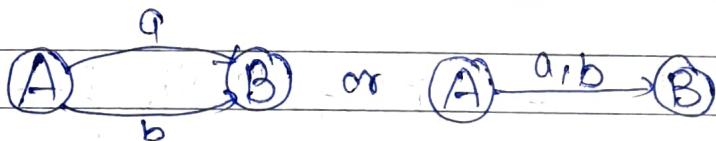
$$\text{also } E + RR^* = R^*$$

$$\therefore R \cdot E = 0^* I^*$$

~~Reg. R. Exp for DFA given~~

Conversion of RE to F.A

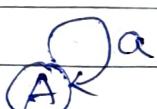
① $(a+b)$



② $(a \cdot b)$



③ a^*



Q) Convert the following Regular Expressions to their Equivalent Finite Automata.

1) $b a^* b$



2) $(a+b)c$



3) $a(bc)^*$

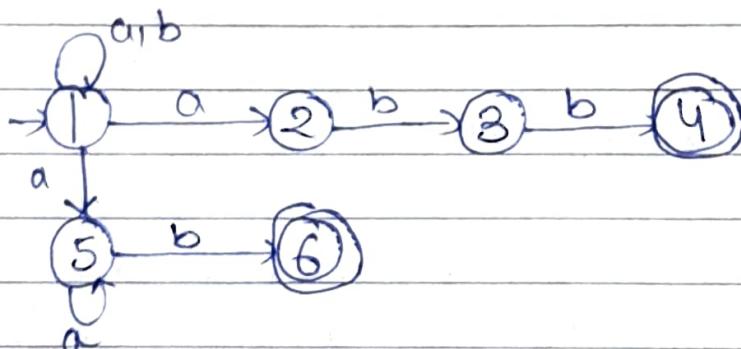
$L = a, abc, abbc, \dots, abcabc$



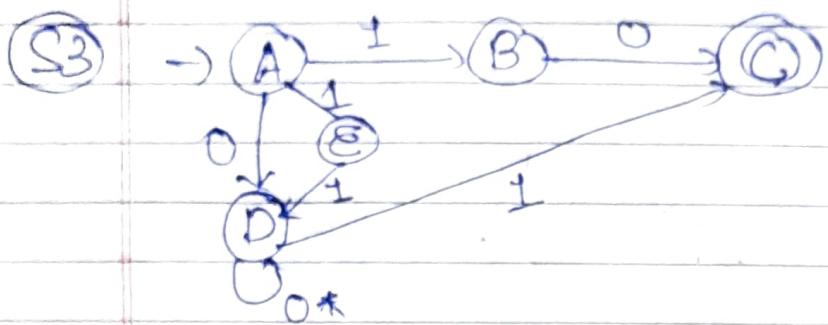
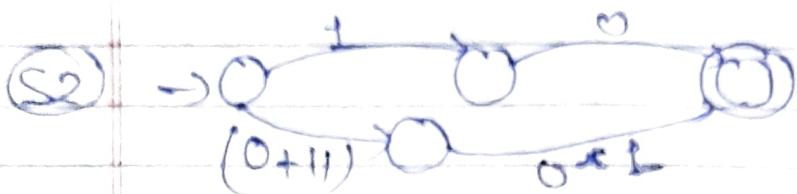
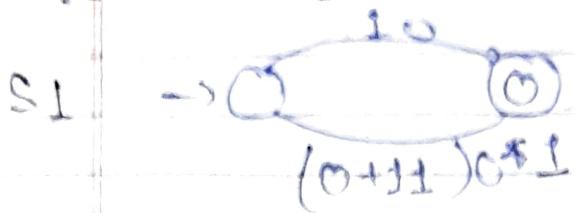
4) $(a|b)^* (abb|a^+ b)$

at least 1 a.

$a^+ = \{a, aa, aaa, \dots\}$
 $a^* = \{\epsilon, a, aa, \dots\}$



Q3. $10 + (0+11)0^*1$



Equivalence of two F.A (Gist of Prev. Equivalence Ques)

Steps to identify equivalence

1) for any pair of states (q_i, q_j) the transition for Input $a \in E$ is defined by $q(q_i, q_j)$ where

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

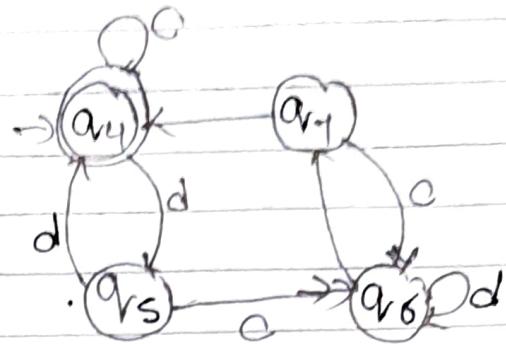
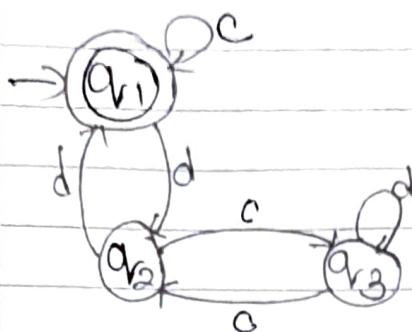
The two automata are not equivalent if for a pair (q_a, q_b) one is intermediate state and other is final state.

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:-

Q.



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 F.S

(q_3, q_6)
I.S I.S

✓

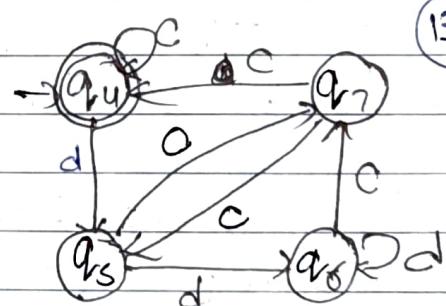
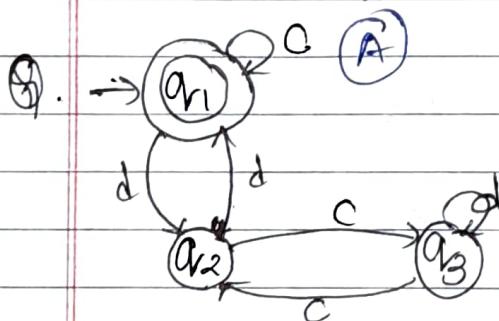
$\{q_2, q_7\}$

(q_3, q_6)
I.S I.S

(q_1, q_4)
F.S F.S

✓

Hence equivalent F.I.A A and B



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 I.S

The given A and B automata are NOT equivalent

Pumping Lemma (for Regular Languages)

- Pumping lemma is used to prove that a language is NOT REGULAR
- It ~~can't~~ 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 ~~$|S| \geq P$~~ $|S| \geq P$
- ③ Divide S into $X Y Z$ Such that i) $|y| > 0$
ii) $|xy| \leq P$

④ Now Show that

$$XY^iz \notin A \text{ for some } i$$

⑤ After proving $XY^iz \notin A$

write it This contradicts our assumption
hence A is not Regular language

Q. Using pumping lemma prove that the language
 $A = \{a^n b^n | n \geq 0\}$ 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 = aaaaaaaaaaaaaaaaaaaaaaa

③ Divide S into XYZ such that i) $|y| > 0$

ii) $|XY| \leq P$, e.g,

Hence S = aaaaaaaaaaaaaaaaaaaaaa,
x y z

④ $S = \underbrace{aaaaaa}_{x}, \underbrace{bbbbbb}_{y}, \underbrace{\dots}_{z}$

$xyz = aaaa aa bbbb bbbb$

Let $|z| = 2$

$\therefore xyz = aaaa aaaa bbbb bbbb$

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

Now

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

Hence $|a| \neq |b| \text{ if } xyz$
 $\therefore xyz \notin A$ for $i \geq 2$

⑤ Thus 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 \therefore F.S.M can't have large memory
 \therefore 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 $S = 0^p 1 0^p 1$

$\therefore S = 000000010000001$

- ③ Divide S into XYZ such that i) $|Y| \geq 0$
ii) $|XY|^i \leq P$

$$\therefore S = \underbrace{00000000}_n \underbrace{1}_{y} \underbrace{00000000}_z$$

or

~~S = 00000010000000~~

④ $\therefore S = \underbrace{00000}_n \underbrace{00}_y \underbrace{10000000}_z$

$$xyz = 00000 \ 000 \ 10000000 \text{ if } i=2$$

1st 2nd

Condition of Lang $\rightarrow |0|=|0|$
in 2 halves. $\rightarrow |1|=|1|$

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

$\therefore xyz \notin A$ if $P=2$

- ⑤ Thus contradicts our assumption hence
A is not a Regular language

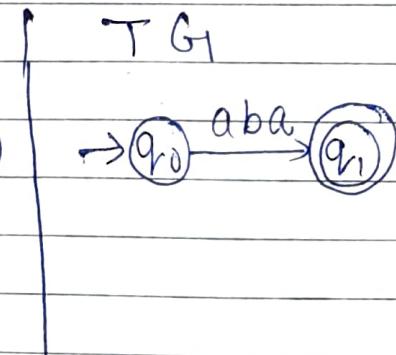
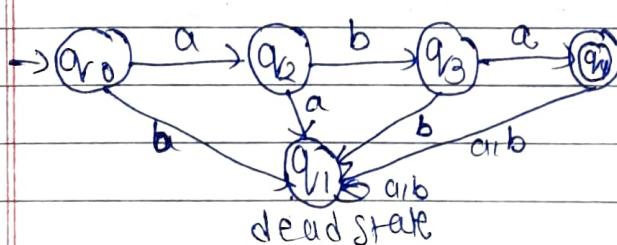
Some missed topics.

Kleen's Theorem

- ① If language can be accepted by F.A (Finite Automata)
then it can also be accepted by T.G (Transition graph)

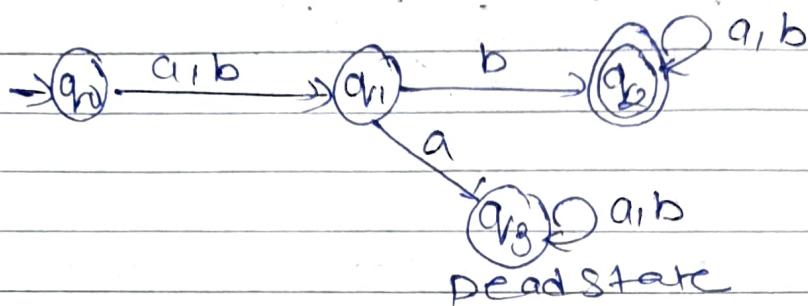
Ex:- Let $S = ab\alpha$

NFA/DFA



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

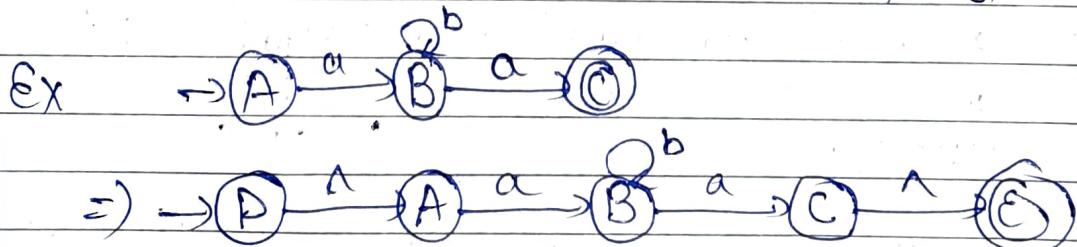
$$\text{Ex: } R.E = (a+b)b(a+b)^*$$



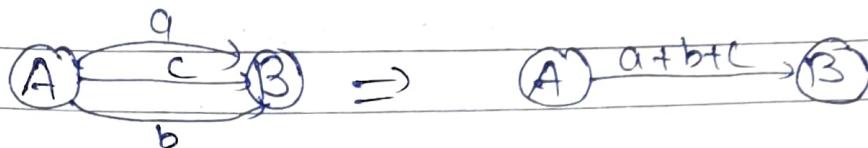
③ If language can be accepted by TG then it can also be expressed by RE

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

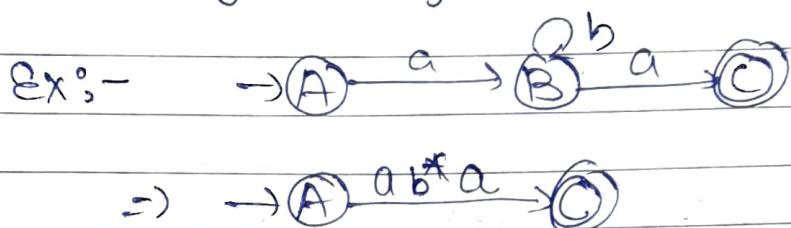
① ~~TA/~~ Introduce a new initial state and final state and connect it with old initial state and final state respectively by the transition labelled as (^) for making the old initial state as Non-initial state Old final state as Non-final state



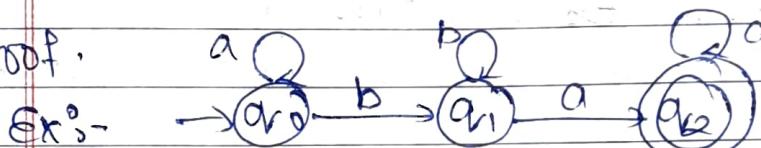
- Q) If a state has an option of (two or more than two) incoming transition edges, then replace all these transition edges with a single (U) transition (unum) transition denoted by (+) symbol.



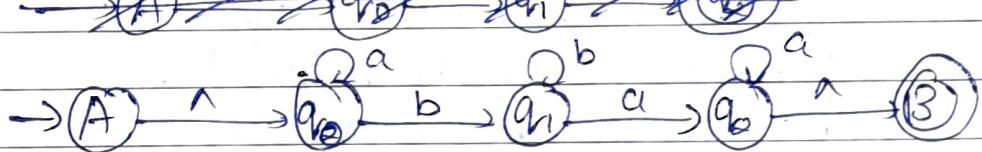
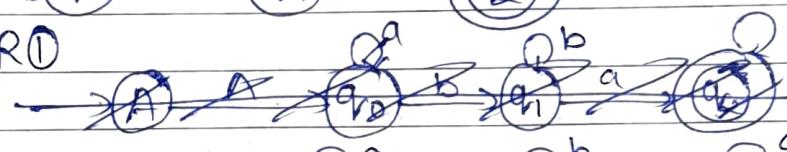
- ③ If 3 states in a TG, are connected in sequence then eliminate the middle state and connect the first state with the third by a single transition.



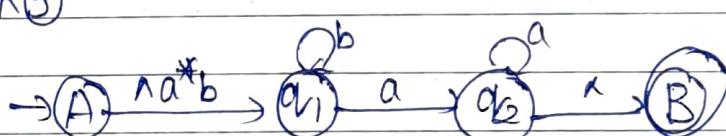
Proof.



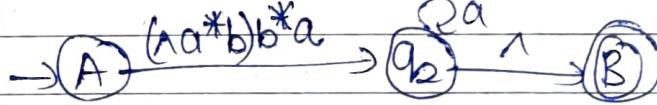
Apply R①



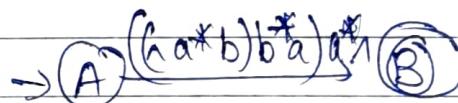
Apply R③



Apply R③



again R③



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

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 state.
also initial and final state can be more than one.
- (3) No dead state
- (4) Multiple transitions 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:- $L = \{a, aa, aab\} \rightarrow$ no state is repeated
or

$L = aabb, bbaa, abbaab, abbbabbabb \dots$
 \rightarrow a state has been repeated

If. String w

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

then the transitions 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 decisions 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 lang?
 - 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 not
 \rightarrow 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 other states.
- (4) Remove the edges of next marked state ~~edges~~ and if next marked state edges connected with another state then also mark 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.

Q) Decide whether given R.E / FA represents finite or infinite language?

Case 1: we have given R.E, we have to check is given R.E represents finite or infinite 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.

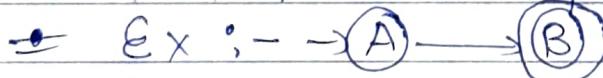
Ex:- $R.E = (a+^*)a(b+^*)b$

- If we have kleen star other than null, we can say R.E must represent infinite language.

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 language.

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



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

