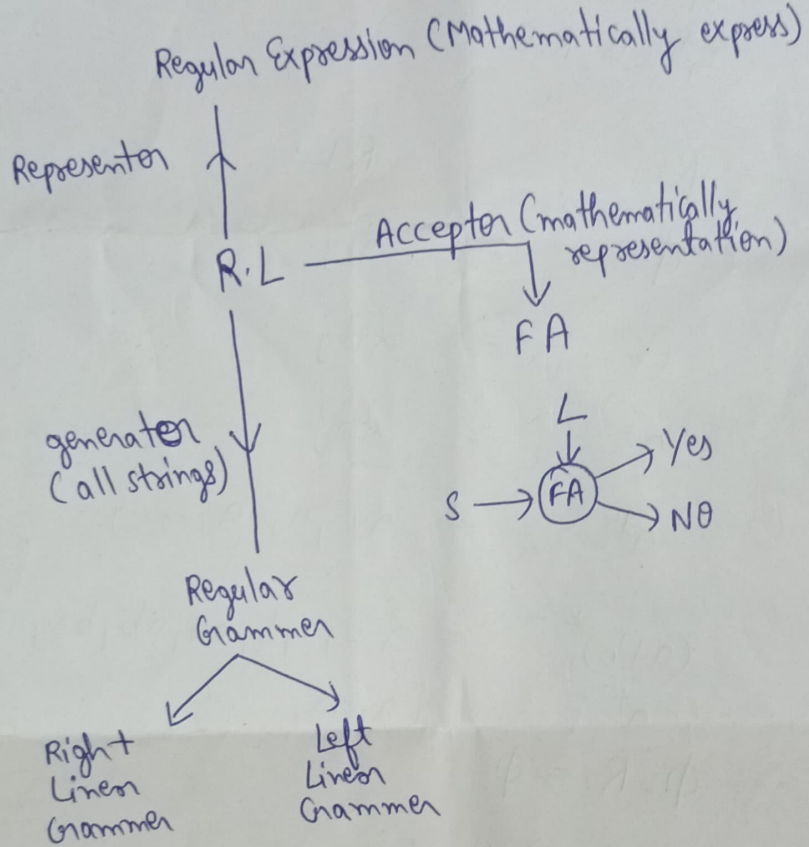
Unit-2 Regular Language

→



→ **Regular Expression** –

- representation of language accepted by FA.
- has 3 operator –
  - (i) + (union)
  - (ii) . (concat)
  - (iii) * (Kleene closure)
- Primitive Regular Expression –

→ ϕ or $\{\}$ empty set

→ ϵ or $\{\epsilon\}$ Null or empty string

- r_1 is R.E and r_2 is R.E

  $r_1 + r_2$ is RE
  $r_1 . r_2$ is RE
  r_1^* is R.E

RE		RL
ϕ	=	$\{\}$
ϵ	=	$\{\epsilon\}$
a	=	$\{a\}$
a^*	=	$\{\epsilon, a, aa, \ldots\}$
$(a+b)^*$	=	$\{\epsilon, a, b, aa, ab, ba, \ldots\}$
a^+	=	$\{a, aa, \ldots\}$

- $\phi . R \Rightarrow \phi$

  $\epsilon . R \Rightarrow R$

Example1

$\Sigma = \{a, b\}$ L_1 = strings of of length exactly 2

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

- $RE = aa + ab + ba + bb$
  $= a(a+b) + b(a+b)$
  $= (a+b)(a+b)$

for length n,
$(a+b) \ldots\ldots$ n times

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

Q L_1 = length is exactly 2

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

Q L_1 = length is atleast 2

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

Q length atmost 2

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

Q Even length strings

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

Just for understanding,

$(a+b)^{2n}$

$(a+b)^{2*}$

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

Q odd length strings

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

Q length is divisibly by 3

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

Q length $\cong 2 \bmod 3$

$(a+b)^{3n+2}$

$(a+b)^{3*} \cdot (a+b)^2$

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

Q $n_a(w) = 2$

$b^* a b^* a^*$

Q atleast two a

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

Q atmost two a

$b^* + b^* a b^* + b^* a b^* a b^*$

or

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

Q a's are even

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

Q starts with a

$a(a+b)^*$

Q ends with a

$(a+b)^* a$

Q contain a

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

Q start & end with different symbols.

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

Q some symbol.

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

Q no 2 a's together

1st $(b+ab)^* + (b+ab)^*a$

2nd $(b+ba)^* + a(b+ba)^*$

Q no 2a's, no 2b's should come together

$L = \{\epsilon, a, b, ab, ba, aba, bab, \ldots\}$

starts with	ends with	
a	a	$\rightarrow (ab)^*a$ or $a(ba)^*$
a	b	$\rightarrow (ab)^*$ or $a(ba)^*b$
b	a	$\rightarrow (ba)^*$ or $b(ab)^*a$
b	b	$\rightarrow (ba)^*b$ or $b(ab)^*$

Take any one and apply union,

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

Identities of RE

- $\phi + R = R = R + \phi$
- $\phi . R = R . \phi = \phi$
- $\epsilon . R = R . \epsilon = R$
- $\epsilon^* = \epsilon$
- $\epsilon + RR^* = R^*$
- $\boxed{\phi^* = \epsilon}$

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

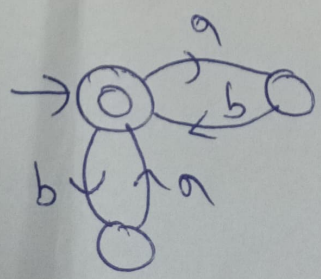
~~Conversion of RE to FA~~

Kleen's Theorem –

A language is said to be regular if it can be represented by using a FA or if a RE can be written for it, so every RE corresponding to the language, a FA can be generated.

Conversion of R.E to FA

- ϕ =
- ϵ =
- a =
- $a+b$ =
- $a.b$ =
- ab^* =
- $(ab)^*$ =
- $(ab+ba)^*$ →

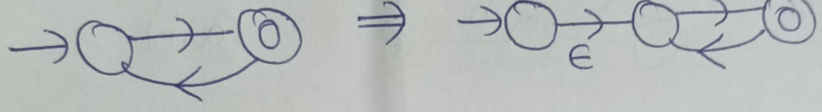


Convert FA to RE -

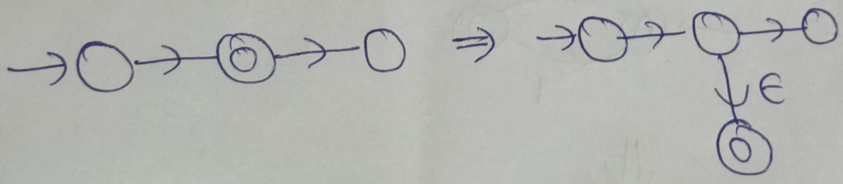
1) State Elimination Method (for ε-NFA, NFA, DFA)

Method -

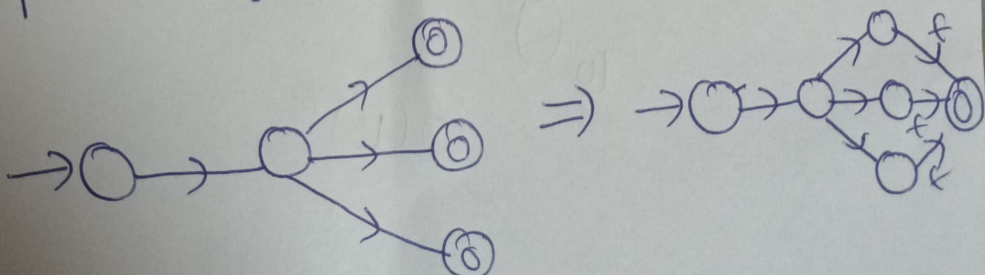
1) Initial state should not have incoming edge (If u see incoming edge, create new initial state with ε move)



2) There should be no outgoing edge from final state (If u see create new final state with ε move)

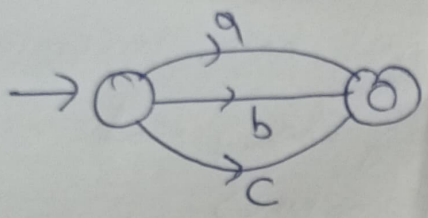


3) If there is more than one final state, convert it into one final state. (Make new final state with ε move from previous final state).



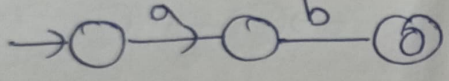
4) other than initial and final state, eliminate state one by one (in any order), but all path present initially should remain intact i.e same effect.

Ex1



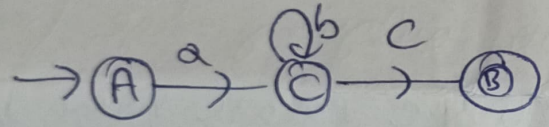
(a+b+c)

Ex2



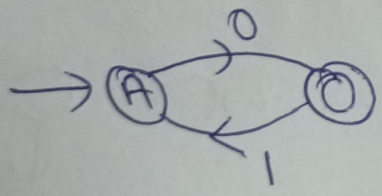
(a.b)

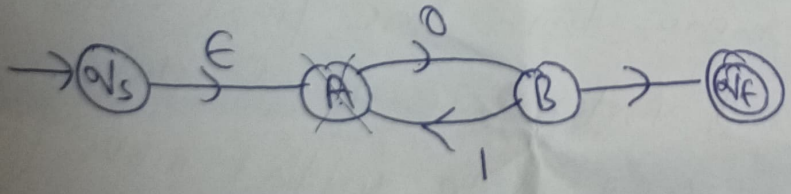
Ex3

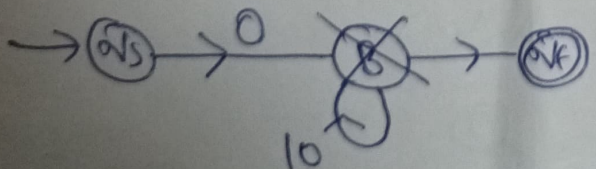


ab* c

Ex4

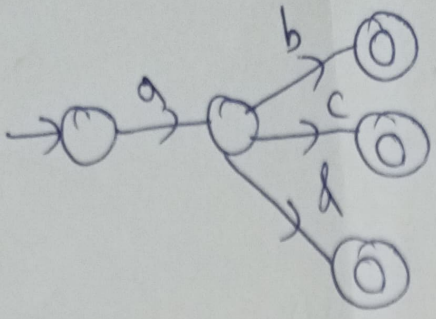






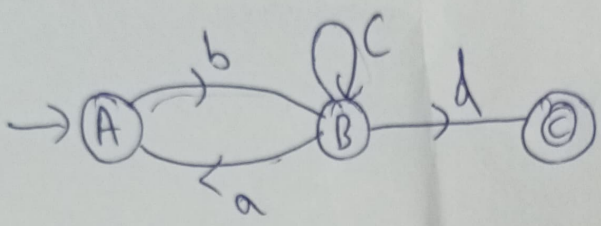
0(10)*

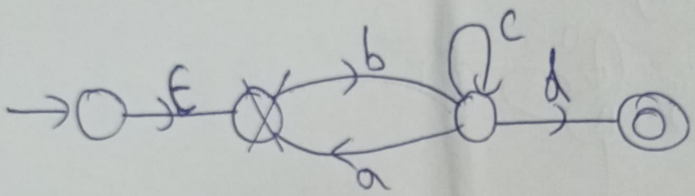
Q

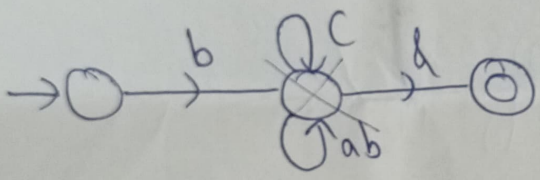


$a.(b+c+d)$

Q

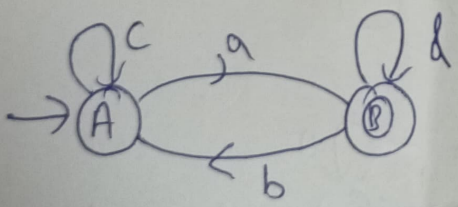


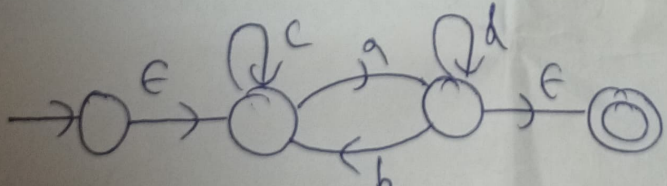


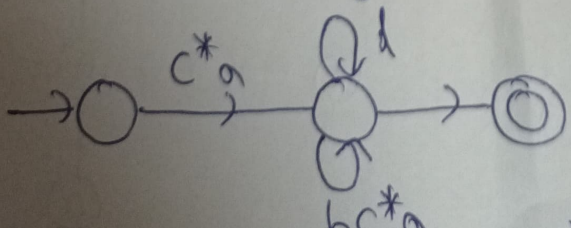


$b(c+ab)^* d$

Q

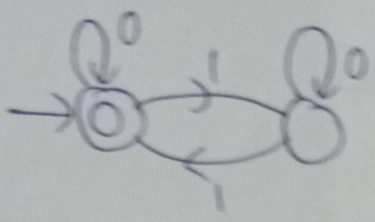


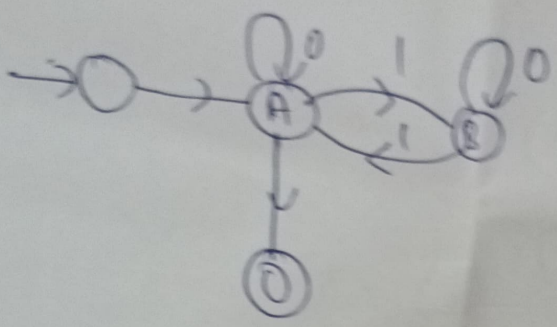




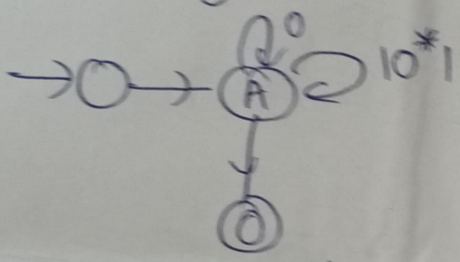
$c^* a (d + bc^* a)^*$

Q



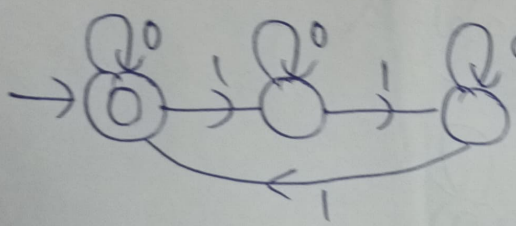


⇓

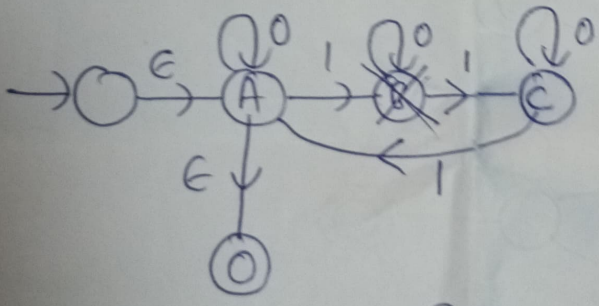


RE = $(0 + 10^*1)^*$

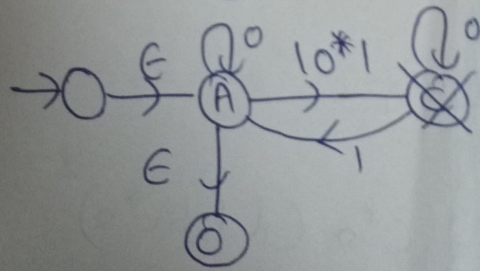
Q



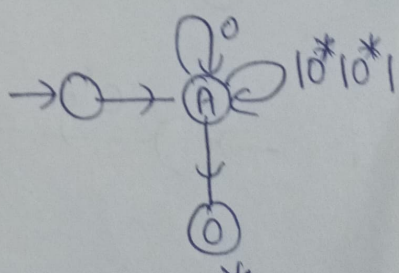
①



②



③



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

FA to RE (Arden's theorem) –

- works for NFA, DFA (not on ∈-NFA)

Arden's Lemma / Rule –

If R, P, Q are regular expressions over Σ, then the following equation given by,

$R = Q + RP$ has a unique soln

$$\Downarrow$$

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

Proof:

$R = Q + RP$ —— ①

Now, replace R by $R = Q + RP$,

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

$= Q + QP + RP^2$

Again, replace R by $R = Q + RP$,

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

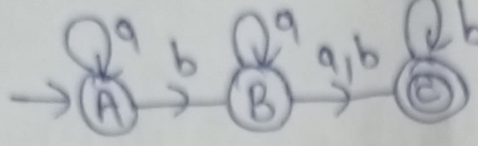
$= Q + QP + QP^2 + RP^3$

$= Q + QP + QP^2 + \ldots + QP^n + \ldots$

$= Q(\epsilon + P + P^2 + \ldots)$

$= QP^*$, Hence Proved

Example 1



I. How do 9 arrive at A

$A = \epsilon + Aa$ —— (1)

Similarly,

$B = Ab + Ba$ —— (2)

$C = Ba + Bb + Cb = B(a+b) + Cb$ —— (3)

Solving using Arden's lema,

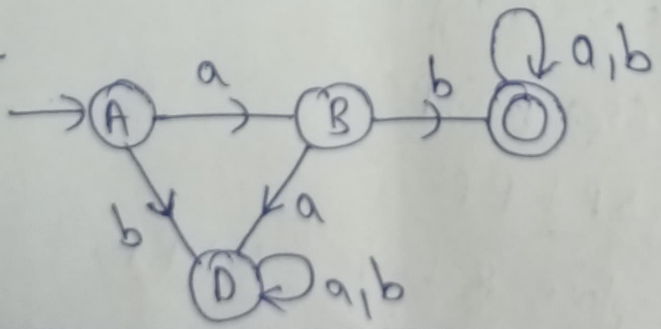
$A = \epsilon a^* = a^*$

$B = Aba^* = a^*ba^*$

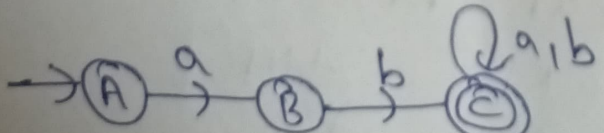
$C = B(a+b)b^*$

$= a^*ba^*(a+b)b^*$ Ans

Example 2



If dead state is there, ignore it



$A = \epsilon$

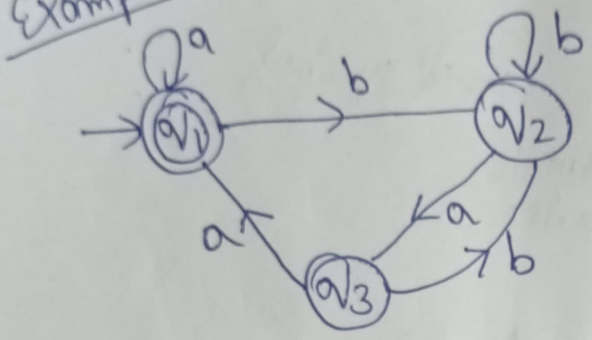
$B = Aa = a$

$C = Bb + C(a+b)$

$C = Bb(a+b)^*$

$= ab(a+b)^*$

Example 3



$q_1 = \epsilon + q_1a + q_3a$ — (I)

$q_2 = q_1b + q_3b + q_2b$ — (II)

$q_3 = q_2a$ — (III)

(III) in (II),

$q_2 = q_1b + q_2b + q_2ab$

$= q_1b(b+ab)^*$

$q_1 = \epsilon + q_1a + q_2aa$

$= \epsilon + q_1a + q_1b(b+ab)^*aa$

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

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

Grammer →

- it does not tell whether string is in the language or not.
- generates entire language.
- represented by,

(V, T, P, S)

- V → vertices
- T → terminals
- P → production rules
- S → start symbol

$S \to aSB$

$S \to aB$

$B \to b$

$V = \{S, B\}$

$T = \{a, b\}$

$S = \{S\}$

$P = \{ S \to aSB,\ S \to aB,\ B \to b \}$

Derivation of aabb –

$S \Rightarrow aSB$

$\Rightarrow aaBB$

$\Rightarrow aabB$

$\Rightarrow aabb$

(leftmost derivation)

⇓

Start replacing leftmost symbol first

$S \Rightarrow aSB$

$\Rightarrow aSb$

$\Rightarrow aaBb$

$\Rightarrow aabb$

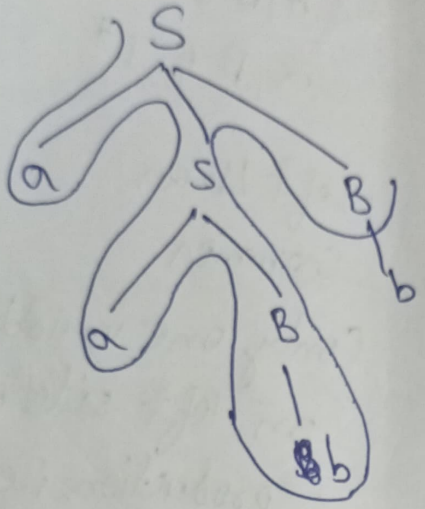
(rightmost derivation)

⇓

Start replacing rightmost symbol first

Derivation Tree / Parse Tree –

- represents syntactic structure of a string according to some context-free grammer.



aabb

Sequential form / Sentential form

- intermediate string on the way of derivation.

$S \rightarrow aSB$

$SB \rightarrow aB$

$B \rightarrow b$

$S \Rightarrow aSB$
$\Rightarrow aaBB$
$\Rightarrow aabB$
$\Rightarrow aabb$

} sentential form

Type 3 or Regular Grammer –

$A \to \alpha B / \beta$

$A, B \in V$

$\alpha, \beta \in T^*$

Right linear Grammer

(only one variable on right side of production i.e extreme right)

$A \to B\alpha / \beta$

$A, B \in V$

$\alpha, \beta \in T^*$

Left Linear Grammer

(only one variable on left side of production i.e extreme left)

Example1

$A \to aB / a$ RLG

Example2

$A \to Ba / a$

$B \to Ba / Bb / a / b$ LLG

Example3

$A \to Ba / a$ ✗

$B \to aB / a$

Note- cannot be combination of RLG and LLG, it should be either RLG or LLG but not both

Check whether grammer is regular or not -

1) If given grammer contain 2 or more than two non-terminal on right side of production, not regular.
2) cannot be both right linear or left linear.

Given a regular language,

1) can have more than one grammer which may or may not be regular.
2) So, we can give atleast one regular grammer for every regular language.

Examples:-

Ex1 $L = \{ab, ab, ba, bb\}$

$S \to AA$

$A \to a/b$

$\underbrace{(a+b)}_{A}\underbrace{(a+b)}_{A}$

Ex2 $a^n / n \geq 0$

$L = \{\epsilon, a, aa, \ldots\}$

$A \to aA/\epsilon$ or $A \to Aa/\epsilon$

Ex3 $(a+b)^*$

$S \to aS/bS/\epsilon$

Ex4 length atleast '2'

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

$S \to AAB$

$B \to aB/bB/\epsilon$

$A \to a/b$

Ex 5 atmost two

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

$S \to AA$
$A \to a/b/\epsilon$

Ex 6 $a(a+b)^* b + b(a+b)^* a$

$S \to aAb/bAa$
$A \to aA/bA/\epsilon$

Ex 7 same symbol

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

$S \to aAa/bAb/a/b/\epsilon$
$A \to aA/bA/\epsilon$

Ex 8 Even length string

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

$S \to BS/\epsilon$
$B \to AA$
$A \to a/b$

Q $a^n b^m / n, m \geq 1$

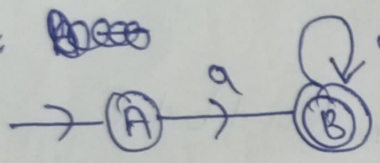
$S \to AB$
$A \to aA / a$
$B \to bB / b$

Conversion

① FA $\longrightarrow$ R.G

(Initial state in FA is initial symbol in RG)

Q



$A \to aB$
$B \to aB / bB / \epsilon$ $\longrightarrow$ RLG
(L)

⇓ Reverse

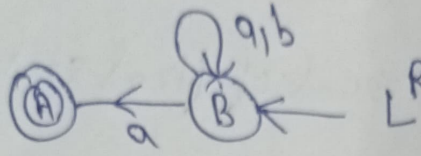
$A \to Ba$
$B \to Ba / Bb / \epsilon$
(L^R)

I.M.P

FA $\longrightarrow$ RLG $\xrightarrow{\text{Reverse}}$ LLG
(L) (L) (L^R)

So, to get same Language L for LLG

FA (L) $\xrightarrow{R}$ FA (L^R) $\longrightarrow$ RLG (L^R) $\xrightarrow{R}$ LLG (L^R)^R = L



$B \rightarrow aB/bB/aA$

$A \rightarrow \epsilon$

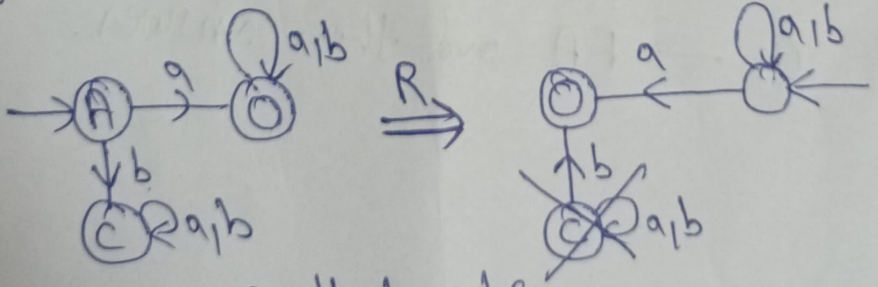
$\downarrow R$

$B \rightarrow Ba/Bb/Aa$

$A \rightarrow \epsilon$ LLG

For Reversal of FA –

1) Draw the FA first.
2) Make initial as final and final as initial.
3) Reverse the transition, loop will remain same.
4) Remove non-reachable state.



Resultant can be

L^R < NFA or DFA

(12)

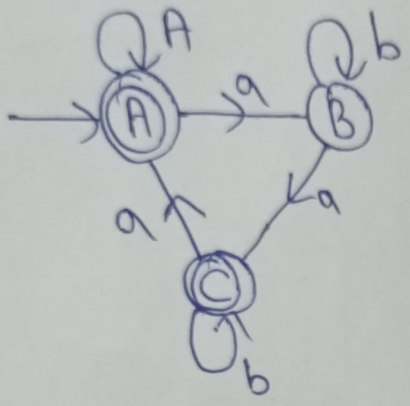
(2) RG → FA

RLG → FA

A → aB | bA | b → end of final state

B → aC | bB

C → aA | bC | a → end of final state



LLG → FA

$$LLG \xrightarrow{R} RLG \rightarrow FA \xrightarrow{R} FA$$

LLG (L) → RLG (L^R) → FA (L^R) → FA $(L^R)^R = L$

Question Possible –

Give Grammer and ask what language?

G → FA and then answer.

- Is L regular or not?

1) $a^n \mid n \geq 1$ ✓

2) $a^n b^m \mid n, m \geq 1$ ✓

3) $a^n b^n \mid n \leq 10^{10^{10}}$ ✓

4) $a^n b^n \mid n \geq 1$ X

5) $ww^R \mid |w| = 2 \quad \Sigma \in \{a, b\}$

✓

6) $ww^R \mid w \in (a, b)^*$ X

7) $a^n b^m c^k \mid n, m, k \geq 1$ ✓

8) $a^i b^{2j} \mid i, j \geq 1$ ✓

9) $a^n \mid n$ is even ✓

10) $a^n \mid n$ is odd ✓

11) $a^n \mid n$ is prime X

12) a^{n^2} X

13) a^{2^n} X

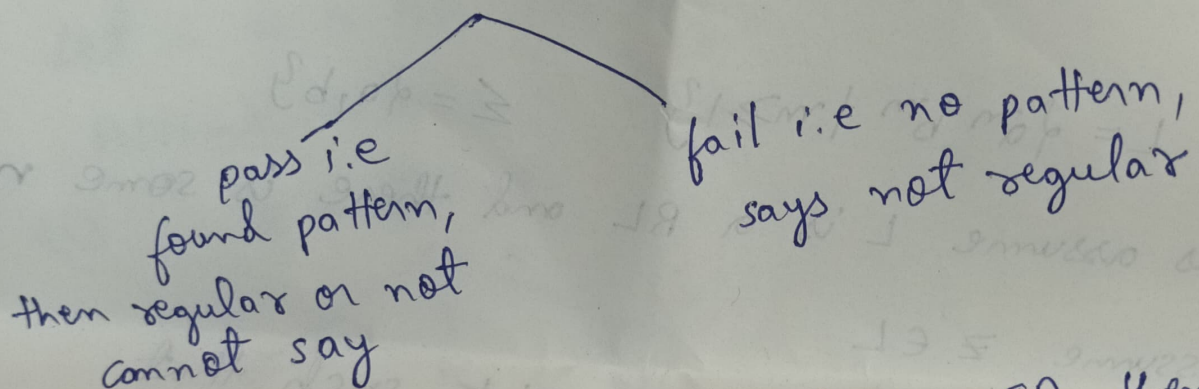
14) $a^i b^{j^2}$ X

Pumping Lemma for Regular Languages –

→ finite language $\xrightarrow{\text{definitely}}$ Regular

→ Infinite language $\xrightarrow{\text{may or may not be}}$ Regular

→ Pumping Lemma is a negative test.



A infinite language is accepted by FA then there has to be a loop inside automata and inside a loop there must be a pattern which repeats to get all other strings in language.

Theorem –

for any RL, L $\exists$ an integer n such that

$\forall z \in L$ and $|z| \geq n$

and (i) $z = uvw$

(ii) $|uv| \leq |z|$

(iii) $|v| \geq 1$ or $|v| \neq 0$

then it says $uv^i w \in L \quad \forall i \geq 0$

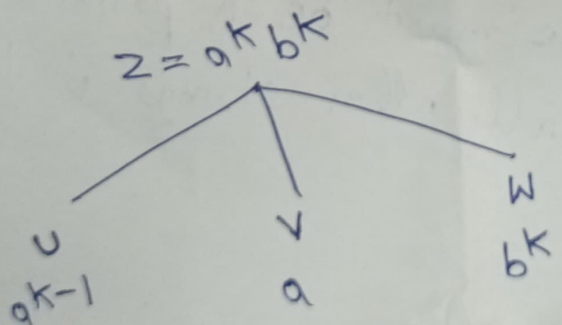
Ex1 $L = \{a^m b^m \mid m \geq 1\}$ $\Sigma = \{a, b\}$

Let us assume L is a RL and there is some n

Assume $z \in L$

$z = a^k b^k$ choose k such that $|a^k b^k|$ which $2k \geq n$

(i)



(ii) $|uv| = |a^{k-1}a| = k$

$|z| = 2k$

$k \leq 2k$

(iii) $|v| \geq 1$

hence $uv^iw \in L \quad \forall i \geq 0$

$i=2 : a^{k-1} a^2 b^k = a^{k+1} b^k \notin L$

therefore our assumption is wrong
and L is not RL.

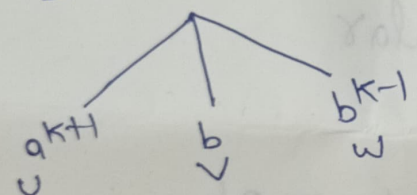
Ex2

$L = \{a^m b^p \mid m > p\}$

Let L is a RL and there is some n

Assume $z \in L$

(i) $z = a^{k+1} b^k$



(ii) $|uv| \leq |z|$

$|a^{k+1} b| \leq |a^{k+1} b^k|$

$k+2 \leq 2k+1$

(iii) $|v| \neq 0$

hence $uv^iw \in L \quad \forall i \geq 0$

$i=1 \quad a^{k+1} b^1 b^{k-1}$ ✓

$i=2 \quad a^{k+1} b^2 b^{k-1} = a^{k+1} b^{k+1} \notin L$

therefore not regular

Special Case of pumping Lemma –

If i/p alphabet 'Σ' is singleton set & Language L is defined over Σ, then L is RL iff lengths of strings in L are in AP.

ex- $L = \{a^{n!} \mid n \geq 1\}$

$= \{a, a^2, a^6, a^{24}, \ldots\}$

not regular

Ex2 $L = \{a^{2n+1} \mid n \geq 0\}$

$= \{a, a^3, a^5, a^7, \ldots\}$

regular

Pigeonhole Principle –

→ We can phrase many counting problem in terms of ordered or unordered arrangements of objects of a set.

→ it is an elementary but important combinatorial tool that can be used to solve a variety of counting problems associated with arrangements.

→ Principle – If a lot of pigeons fly into not too many pigeonholes, then at least one pigeonhole will be occupied by two or more pigeons.

Theorem 1 If (n+1) objects are put into n boxes then atleast one box contains two or more of the objects.

Proof –

If each of the n boxes contains atmost one of objects, then total no of objects is at most n. Since, we start with (n+1) objects, so we have still one more object to put in some box. So, clearly, the box we put this (n+1)th

object (or last one) will contain two objects.
Proved

Limitation

→ it doesn't identify which box contains two or more objects.

objects — Pigeons

Boxes — Pigeonholes

Coloring Problem

Consider coloring of objects with 'n' colors
The pigeonhole principle asserts that if (n+1) objects are colored with 'n' colors then atleast two objects have some color.

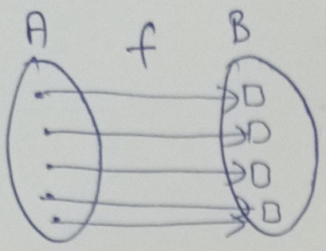
color - pigeonhole
objects - pigeon

Q There are n-married couples. How many of 2n people must be selected in order to guarantee that one has selected a married couple?

MC → Pigeonhole
H/W → Pigeon

n+1

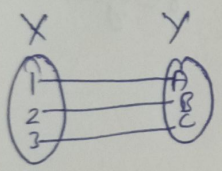
→



pigeon pigeon-hole

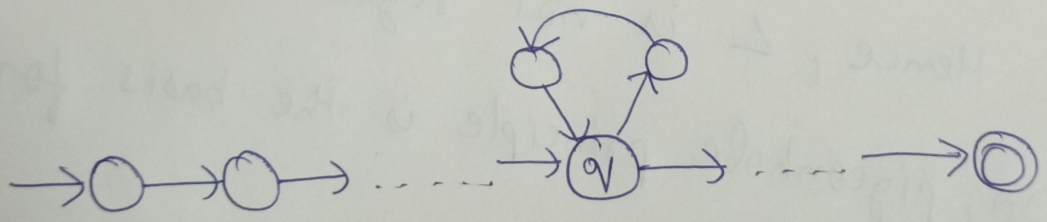
A → Domain
B → Co-Domain
Range is set of all images of A so, subset of B

- function is a non-injective surjective function (onto) but not bijection
- every element of codomain is image of atleast one element of its domain.
- function f may map one or more elements of A to some element of Y.



injective surjective func^{n} (bijection)

→



$L = \{0^n 1^n \mid n \geq 1\}$ is regular?

If string w has length $|w| \geq n$ (# states in DFA) then, from pigeonhole principle:

a state is repeated in walk

state → Holes
strings → pigeons

Let us take two string,

$0^i 1^i$ & $0^j 1^i$

such that $i \neq j$

for some value of i, 0^i will end in state q and for some value of j, 0^j will end in state q

Now, $0^i 1^i$ ends in final state and it belongs to L

but $0^i 1^j$ also ends in final state and it does not belong to L

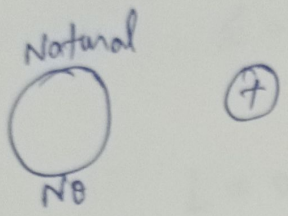
So, our assumption that L is regular is false.

Hence, L is not regular.

So, pigeonhole principle is the basis for pumping lemma.

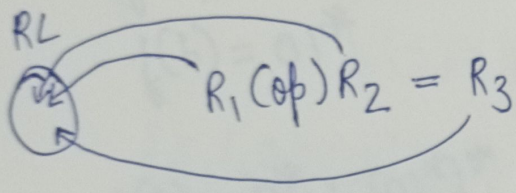
Closure Properties of RL -

closed under operation



⊕

If i take two natural no, apply + operation then result is also natural no.



RL are closed under that operation

1) **Union** - Take R.E of both L_1 & L_2

$L_1 \cup L_2$

$\downarrow \quad \downarrow$

$RE_1 + R.E_2 = RE_2$ ≡

2) **Concatenate** -

$L_1 . L_2$

$\downarrow$

$\boxed{R_1 . R_2} \longrightarrow R_3$ ≡

3) Kleene closure -

L_1^*

↓

R_1^* =

4) Complement

$\overline{L_1} = \Sigma^* - L_1$

If L_1 is regular

↓

DFA

↓ complement

$\overline{L_1}$ → Regular

5) Intersection

$L_1 \cap L_2$

$= \overline{\overline{L_1} \cup \overline{L_2}}$

$\overline{L_1}$ → Regular, $\overline{L_2}$ → Regular

$\overline{L_1} \cup \overline{L_2}$ → regular

$\overline{\overline{L_1} \cup \overline{L_2}}$ → regular

So, $L_1 \cap L_2$ → regular

6) Substitution - means for every symbol of language is replace by other language

ex- $\Sigma = \{a, b\}$

$L = a + b^*$

$f(a) = 0^*$ (for every symbol a, replace by language 0^*)

$f(b) = 01^*$

$f(L) = 0^* + (01^*)^* \rightarrow RE$

7) Difference & Reversal -

Difference

$L_1 - L_2$

$L_1 \cap \overline{L_2}$ → closed

→ closed

Reversal

$L \rightarrow DFA \xrightarrow{R} L^R$ (DFA or NFA possible) so closed

=

(8) Homomorphism

→ for every symbol in input, if you substitute a string from some other alphabet, then it is called homomorphism.

→ New string after substitution is called homomorphic image of string.

function $h: \Sigma \longrightarrow \Gamma^*$

Ex- $\Sigma = \{a, b\}$ $\Gamma = \{0, 1, 2\}$ other i/p

$h(a) = 01$
$h(b) = 112$

(for every i/p a, replace by 01)

$h(aba) = 011201$

$L = a^* b = (01)^* (112)$, so homorphic image of language

2) Inverse Homomorphism

→ If we substitute back original alphabet in place of some othe alphabet, it is inverse homomorphism.

Ex Let $\Sigma = \{0,1,2\}$ $\Delta = \{a,b\}$

Define 'h' by $h(0) = a$ $h(1) = ab$

$h(2) = ba$

~~$L_1 = \{ababa\}$~~

~~$h^{-1}(L_1) = \{110, 022, 102\}$~~

$L_2 = a(ba)^* = \{a, aba, ababa, \cdots\}$

$h^{-1}(L_2) = \{0, 02, 10, 110, 022, 102, \cdots\}$

IMP Let $\Sigma = \{0,1\}$ and $\Delta = \{a,b\}$

Define 'h' by $h(0) = aa$ $\quad h(1) = aba$

$L = (ab + ba)^* a$

$= \{a, aba, baa, ababa, abbaa, \ldots\}$

$h^{-1}(L) = \{1\}$

$h(h^{-1}(L)) = \{aba\}$

$\neq L$

$\therefore \boxed{h(h^{-1}(L)) \subseteq L}$

(we may or may not get entire language back)

10) Right quotient - closed

L_1 & L_2 be language on some alphabet

$L_1/L_2 = \{x : xy \in L_1 \text{ for some } y \in L_2\}$

Ex1 $L_1 = \{01, 001, 101, 0001, 1101\}$

$L_2 = \{01\}$

$L_1/L_2 = \{\epsilon, 0, 1, 00, 11\}$

Ex2 $L_1 = 10^*1$ $L_2 = 0^*1$

$L_1/L_2 = 10^*$

$L_1 = \{11, 101, 1001, 10001, \ldots\}$

$L_2 = \{1, 01, 001, 0.001, \ldots\}$

$L_1/L_2 = \{1, 10, 100, 1000, \ldots\}$

$= 10^*$

(11) Init operation - (Take every string apply prefix)

Ex $L = \{a, ab, aabb\}$

$Init(L) = \{\epsilon, a, \epsilon, a, ab, \epsilon, a, aa, aab, aabb\}$

$\Downarrow$ set, so write once

$= \{\epsilon, a, ab, aa, aab, aabb\}$

Prefix → set of all string from start

suffix / Postfix ⇒ " " " " end

$RAVI = \{\epsilon, R, RA, RAV, RAVI\}$ → Prefix

$= \{\epsilon, I, VI, AVI, RAVI\}$ → saffix

How to prove?

Construct a DFA and make every state final except the dead state, hence proved

(12) Infinite union – not closed (may or may not be regular always)

(1) $L_1 = \{a^1 b^1\}$

$L_2 = \{a^2 b^2\}$

$L_3 = \{a^3 b^3\}$

$L_1 \cup L_2 \cup L_3 \ldots\ldots$

$\{a^n b^n \mid n \geq 1\}$ ✗

(2) $L_1 = \{a^1\}$

$L_2 = \{a^2\}$

$L_1 \cup L_2 \cup \ldots\ldots$

$= \{a^n \mid n \geq 1\}$ ✓

(13) Superset/subset - not closed

Ex 0^*1^* RL

subset $\rightarrow$ 0^n1^n is not regular

Decidability Problems on RL –

1) Emptiness Problem of FA –

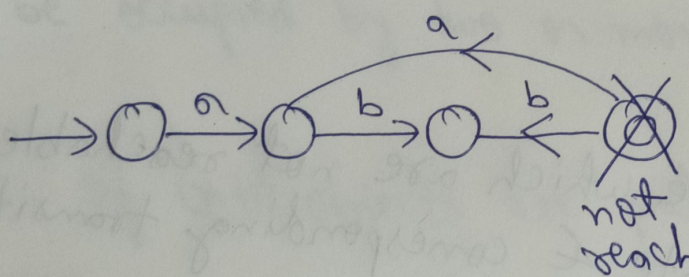
Is it decidable?

Yes

Algorithm –

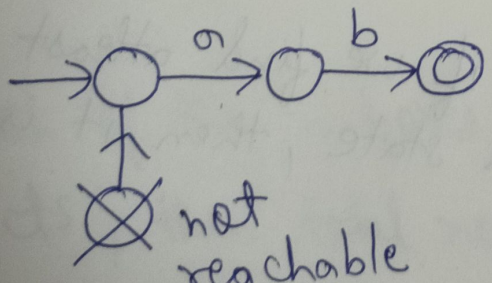
1) Select states which are not reachable from initial state delete those states and corresponding transitions
2) In remaining FA, if we find atleast one final state, then language accepted by given FA is non-empty otherwise empty.

Ex 1



⇒ Empty

Ex 2

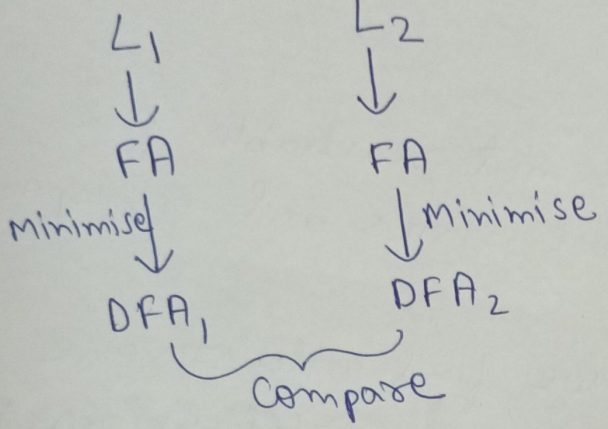


⇒ not empty

2) Is $L_1 = L_2$? decidable

Yes

Algo -



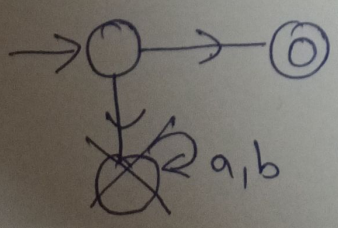
3) Infiniteness Problem of RL ?

Yes

Algo -

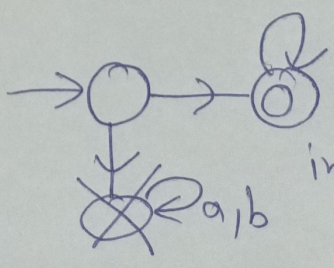
1) Remove state which are not reachable from initial state & corresponding transitions
2) Delete states and " " " from which we cannot reach final states
3) In remaining FA, if we find atleast one loop and one final state, then it is accepting infinite language.

Ex1



finite

Ex2



infinite