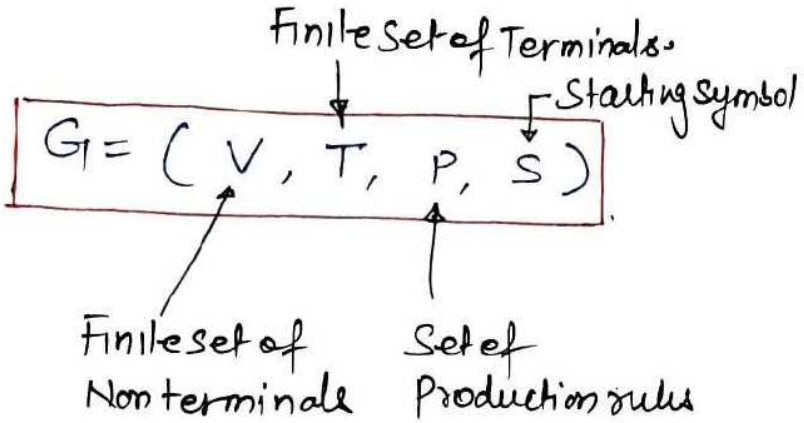
Grammar: is a set of Production rules for strings in a formal language it is represented by 4-tuples

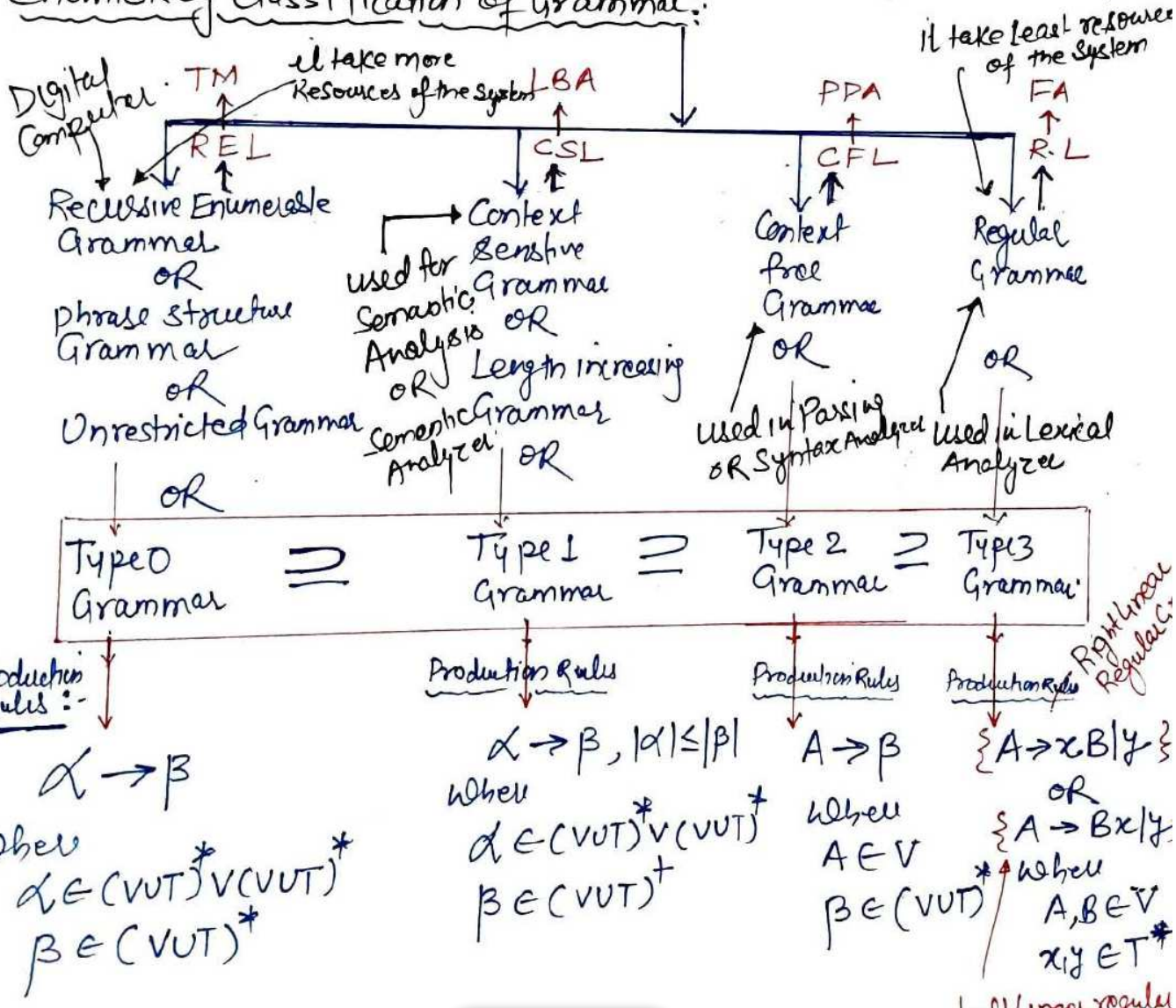


Symbol → Alphabet → Grammar → Strings → Language → Machine → FA, PDA, LBA, TM

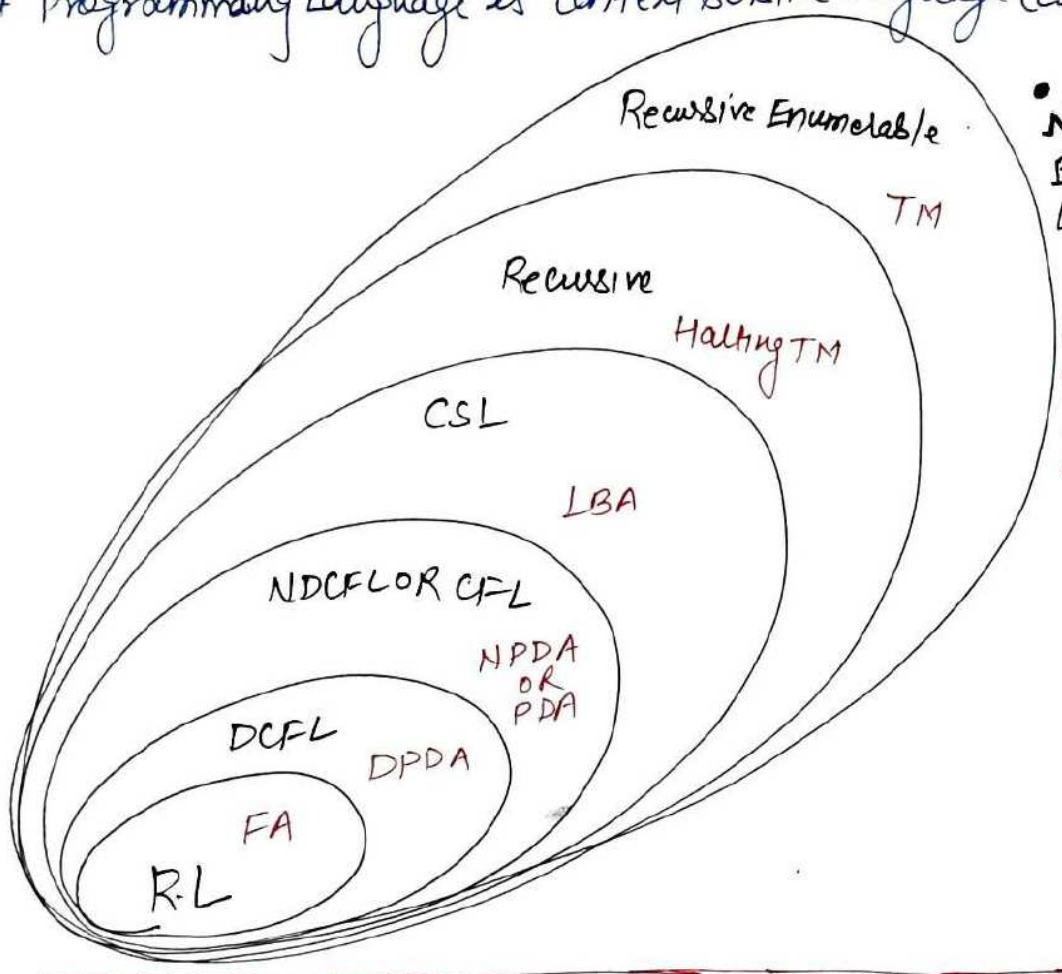
Ex:- $S \to aS \mid aA$
$A \to bA \mid b$

$G = (V, T, P, S) = (\{S, A\}, \{a, b\}, \{S \to aS \mid aA,\ A \to bA \mid b\}, S)$

Chomiskey classification of Grammar:



NOTE: All programming Language Grammar is a Context free Grammar (CFG)
But Programming Language is Context Sensitive Language (CSL)



• Not Recursive Enumerable Language
NO Computer for Not Recursive Enumerable Language

NOTE: $R.L \subseteq DCFL \subseteq CFL \subseteq CSL \subseteq Recursive \subseteq Recursive\ Enumerable$

Grammar Type	Grammar Accepted	Language Accepted	Production Rules	Automata
Type 0	Unrestricted Grammar	Recursive Enumerable Language	$\alpha \to \beta$, where $\alpha \in (V \cup T)^* V (V \cup T)^*$, $\beta \in (V \cup T)^*$	Turing Machine
Type 1	Context Sensitive Grammar OR Length increasing Grammar	Context Sensitive Language	$\alpha \to \beta$, $\lvert\alpha\rvert \le \lvert\beta\rvert$, where $\alpha \in (V \cup T)^* V (V \cup T)^*$, $\beta \in (V \cup T)^+$	Linear Bounded Automata
Type 2	Context free Grammar	Context free Language	$A \to \beta$, where $A \in V$, $\beta \in (V \cup T)^*$	Push down Automata
Type 3	Regular Grammar	Regular Language	$A \to xB/y$ ← Right Linear Regular Grammar OR $A \to Bx/y$ ← Left Linear Regular Grammar, where $A, B \in V$, $x, y \in T^*$	Finite Automata

0 0

Ex!- Identify Production Rules of each Production and Grammar

G_1 = (Overall Grammar Type is Superset of Grammar i.e Type0 ↑ type0)

- $aAb \to ab$ – Type0
- $aAb \to aBCb$ – Type1, Type0
- $AB \to BC$ – Type1, Type0
- $AC \to C$ – Type0
- $AB \to AbBC$ – Type1, Type0
- $A \to ABC$ – Type2, Type1, Type0
- $A \to B$ – Type3, Type2, Type1, Type0
- $A \to abc$ – Type3, Type2, Type1, Type0
- $A \to \varepsilon$ – Type3, Type2, Type0 but not Type1
- $A \to aAb$ – Type2, Type1, Type0
- $A \to abcA$ – Type3, Type2, Type1, Type0

Grammar to Language:

Language generated by G is the set $L(G) = \{w \mid w \in T^*, S \overset{*}{\Rightarrow} w\}$ of all strings generated by G

Ex!. $G = (S \to aS/a)$

$S \Rightarrow a$
$S \Rightarrow aS \Rightarrow aa$
$S \Rightarrow aS \Rightarrow aaS \Rightarrow aaa$
$S \Rightarrow aS \Rightarrow aaS \Rightarrow aaaS \Rightarrow aaaa$.

Language is a collection of strings

$L(G) = \{a, aa, aaa, aaaa, \dots\} = \boxed{\{a^n \mid n \geq 1\}}$ Ans.

Ex!- $G\{S \to aS/aA,\ A \to bA/b\}$ Start symbol S

$S \Rightarrow aA \Rightarrow ab$
$S \Rightarrow aA \Rightarrow abA \Rightarrow abb$
$S \Rightarrow aS \Rightarrow aaA \Rightarrow aab$
$S \Rightarrow aS \Rightarrow aaA \Rightarrow aabb$

$\Rightarrow L(G) = \{ab, ab^2, a^2b, a^2b^2, \dots\}$

$\boxed{L(G) = \{a^m b^n \mid m, n \geq 1\}}$ Ans

Ex:- Find Language $L(G) = ?$ of start Symbol S

$$G_1 \begin{cases} S \to aA/\varepsilon \\ A \to aS/a \end{cases}$$

$S \Rightarrow \varepsilon$

$S \Rightarrow aA \Rightarrow aa$

$S \Rightarrow aA \Rightarrow aaS \Rightarrow aa\varepsilon \Rightarrow aa$

$S \Rightarrow aA \Rightarrow aaS \Rightarrow aaaA \Rightarrow aaaa$

$S \Rightarrow aA \Rightarrow aaS \Rightarrow aaaA \Rightarrow aaaaS \Rightarrow aaaaaA \Rightarrow aaaaaa$

$L(G) = \{\varepsilon, a^2, a^4, a^6, a^8 \dots\}$

$$L(G) = \{a^{2n} \mid n \geq 0\}$$ Ans

Ex:- Find Language if start symbol S, $L(G) = ?$

$$G \begin{cases} S \to aSc/aAc \\ A \to bAc/bc \end{cases}$$

$S \Rightarrow aAc \Rightarrow abcc$

$S \Rightarrow aAc \Rightarrow abAcc \Rightarrow abbccc$

$S \Rightarrow aSc \Rightarrow aaAcc \Rightarrow aabccc$

$S \Rightarrow aSc \Rightarrow aaScc \Rightarrow aaaAccc \Rightarrow aaabcccc$

$L(G) = \{abcc, abbccc, aabccc, aaabcccc, \dots\}$

$$L(G) = \{a^m b^n c^{m+n} \mid m, n \geq 1\}$$ Ans

Ex:- Find Language if startsymbol S, $L(G) = ?$

$$G: \begin{cases} S \to 0SA_1 2/012 \\ 2A_1 \to A_1 2 \\ 1A_1 \to 11 \end{cases}$$

Sol^n: $S \Rightarrow 012$

$S \Rightarrow 0SA_1 2 \Rightarrow 0012A_1 2 \Rightarrow 001A_1 22$

$\Rightarrow 00122$

$S \Rightarrow 0SA_1 2 \Rightarrow 00SA_1 2A_1 2 \Rightarrow \dots$

$\Rightarrow 000111222$

$S \Rightarrow 0SA_1 2 \Rightarrow 00SA_1 2A_1 2 \Rightarrow 000SA_1 2A_1 2A_1 2$

$\dots \Rightarrow 000011112222$

$$L(G) = \{0^n 1^n 2^n \mid n \geq 1\}$$

Ans Any

Ex:- Find Language of start symbol S

$$G\begin{cases} S \to aS \mid aA \\ A \to bA \mid bB \\ B \to cB \mid c \end{cases}$$

Soln

$S \Rightarrow aA \Rightarrow abB \Rightarrow abc$

$S \Rightarrow aA \Rightarrow abA \Rightarrow abbB \Rightarrow abbc$

$S \Rightarrow aS \Rightarrow aaA \Rightarrow aabB \Rightarrow aabc$

$S \Rightarrow aS \Rightarrow aaA \Rightarrow aabA \Rightarrow aabbB \Rightarrow aabbc$

$S \Rightarrow aS \Rightarrow aaS \Rightarrow aaaA \Rightarrow aaabB \Rightarrow aaabcB \Rightarrow aaabcc$

Ans:

$$L(G) = \{a^m b^n c^k \mid m,n,k \geq 1\}$$

Ans

$L(G) = \{abc, ab^2c, a^2bc, a^2b^2c, ab^2c^2 \cdots\}$

Ex:- Find Language of start symbol S

$$G\{S \to aSa \mid bSb \mid c\}$$

$S \Rightarrow c$

$S \Rightarrow aSa \Rightarrow aca$

$S \Rightarrow bSb \Rightarrow bcb$

$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abcba$

$S \Rightarrow bSb \Rightarrow baSab \Rightarrow bacab$

$L(G) = \{c, aca, bcb, abcba, bacab \cdots\}$ = odd Length palindrom

$L(G) = \{wcw^T, w \in (a,b)^*\}$ = odd Length palindrom

Ans

Ex:- Find Language of start Symbol S

$$G = \{S \to aSa \mid bSb \mid \varepsilon\}$$

Even Length palindrom

Ans: Similar to previous Question Just replace c by ε

$L(G) = \{w \cdot \varepsilon w^T, w \in (a,b)^*\} = \{aa, bb, abba, baab \cdots\}$

$L(G) = \{ww^T, w \in (a,b)^+\}$ = Even Length palindrom.

Ans

Ex:- Find Language of Start Symbol S, Let $N_a(w)$ and $N_b(w)$ denote the number of a and b in a string w respectively. The language $L(G) \subseteq (a,b)^+$, generated by G is

$$G\begin{cases} S \to bS \mid aA \mid b \\ A \to bA \mid aB \\ B \to bB \mid aS \mid a \end{cases}$$

a) $\{w \mid N_a(w) > 3N_b(w)\}$

b) $\{w \mid N_a(w) > 3N_a(w)\}$

c) $\{w \mid N_a(w) = 3K, K \in (0,1,2 \cdots)\}$

d) $3K, K \in (0,1,2 \cdots)\}$

0 0

Language to Grammar:

Ex:- $L = \{a^n b^n \mid n \geq 1\}$, Find Grammar.

$L = a^n\ b^n \mid n \geq 1$

G: $S \to aSb \mid ab$ Ans

Ex:- $L = \{a^m b^n c^k \mid m, n, k \geq 1\}$

Start

G: $S \to aS \mid aA$
$A \to bA \mid bB$
$B \to cB \mid c$

Ex:- $L = \{ww^T \mid w \in (a,b)^*\}$

G: $S \to aSa \mid bSb \mid \varepsilon$

Ex:- $L = \{wcw^T \mid w \in (a,b)^*\}$

G: $S \to aSa \mid bSb \mid c$

Ex:- $L = \{w \mid |w| \bmod 3 = 0, w \in (a,b)^*\}$

start

G: $S \to aA \mid bA \mid \varepsilon$
$A \to aB \mid bB$
$B \to aS \mid bS \mid a \mid b$

Ex:- $L = \{a^m b^n c^{m+n} \mid m, n \geq 1\}$

start

G: $S \to aSc \mid aAc$
$A \to bAc \mid bc$

Ex:- $L = \{a^m b^n c^k, k = m+2n, m, n, k \geq 1\}$

Both:

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

OR

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

Start

G: $S \to aSc \mid aAcc$
$A \to bAcc \mid bcc$

Ex:- $L = \{a^n b^{2n} \mid n \geq 1\}$

G: $S \to aSbb \mid abb$

Ex:- $L = \{a^n b^n \cup a^n b^{2n} \mid n \geq 1\}$

start

G: $S \to S_1 \mid S_2$
$S_1 \to aS_1b \mid ab$
$S_2 \to aS_2bb \mid abb$

Ex:- $L = \{a^l b^m c^n \mid l = m \text{ OR } m = n,\ l, m, n \geq 1\}$

OR

$L = \{\underbrace{a^m b^m}_{A}\ \underbrace{c^n}_{B} \cup \underbrace{a^l}_{C}\ \underbrace{b^n c^n}_{D} \mid l, m, n \geq 1\}$

start

G: $S \to S_1 \mid S_2$
$S_1 \to AB$, $S_2 \to CD$
$A \to aAb \mid ab$, $C \to aC \mid a$
$B \to cB \mid c$, $D \to bDc \mid bc$

Ex:- $L = \{a^l b^m c^n \mid l = m \text{ AND } m = n,\ l, m, n \geq 1\}$

OR

start $L = \{a^n b^n c^n \mid n \geq 1\}$

G: $S \to aSBC \mid abc$
$CB \to BC$
$cB \to Bc$
$bB \to bb$
$cC \to cc$

Regular Grammar:- if every Production is of the form

either

$A \to xB \mid y$ → Right Linear Regular Grammar

OR

$A \to Bx \mid y$ → Left Linear Regular Grammar.

where $A, B \in V$

$x, y \in T^*$

NOTE:- $A \to xB \mid Bx \mid y$ is CFG but not Regular Grammar.

Finite Automata to Right Linear Regular Grammar:

Let $FA = (Q, \Sigma, \delta, q_0, F)$ be finite Automata then

Grammar $\{A_0, A_1, A_2 \cdots\}$

$G = (V, T, P, S)$ $\quad S = A_0$

where $V = \{A_0, A_1, A_2 \cdots\}$

$T = \Sigma$

$S = \{A_0\}$

P is defined by the following rules

P: R_1: if $\delta(q_i, a) = q_j$, $a \in \Sigma$ Then Production in Grammar $A_i \to a A_j$

R_2: if $q_i \in F$ Then Production in Grammar $A_i \to \varepsilon$

Ex:-

Ans

{Right Linear Regular Grammar: start symbol}

$\{A \to aB$

$B \to aB \mid bB \mid \varepsilon\}$

$FA = (Q, \Sigma, \delta, q_0, F)$

$FA = (\{A, B\}, \{a, b\}, \delta, A, B)$

$G = (V, T, P, S)$

where $V = \{A, B\}$

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

$S = \{A\}$

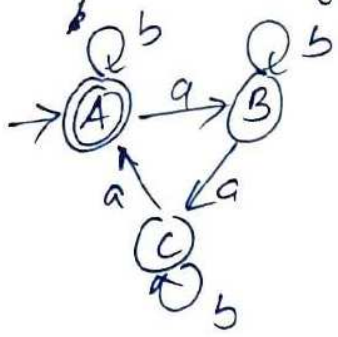
P: R_1: $\delta(A, a) = B$, $A \to aB$

$\delta(B, a) = B$, $B \to aB$

$\delta(B, b) = B$, $B \to bB$

R_2: $B \in F$, $B \to \varepsilon$

Ex:- Convert FA to Right Linear Regular Grammar



Sol^n:- Given $\delta(A,a)=B$, $\delta(A,b)=A$, $\delta(B,a)=C$, $\delta(B,b)=B$, $\delta(C,a)=A$, $\delta(C,b)=C$

Apply Rule 1: $A \to aB$, $A \to bA$, $B \to aC$, $B \to bB$, $C \to aA$, $C \to bC$

Rule 2: $A \in F$, Then $A \to \varepsilon$

Right Linear Regular Grammar (RLRG):

$$G_1 = \left\{ \begin{array}{l} A \to aB \mid bA \mid \varepsilon \\ B \to aC \mid bB \\ C \to aA \mid bC \end{array} \right\} \text{ Ans}$$

Finite Automata to Left Linear Regular Grammar (LLRG):

But not for Given

FA = (L) → (A) —a→ ((B)) ⟲ a, b

RLRG: $G_1 \left\{ \begin{array}{l} A \to aB \\ B \to aB \mid bB \mid \varepsilon \end{array} \right\} \Rightarrow L(G_1) = a(a+b)^*$

|| Reverse

LLRG: $G_2 \left\{ \begin{array}{l} A \to Ba \\ B \to Ba \mid Bb \mid \varepsilon \end{array} \right\} \Rightarrow L(G_2) = (a+b)^* a$

$L(G_1) \neq L(G_2)$

LLRG for ((A)) ←a— (B) ⟲ a, b = $(FA)^T$

$(FA)_{(L)} \to RLRG_{(L)} \xrightarrow{\text{Reverse}} \cancel{LLRG}_{(L^R)}$

NOTE:-

$$FA_{(1)}(L) \xrightarrow{\text{Reverse}} FA_{(2)}(L^R) \to RLRG_{(3)}(L^R) \xrightarrow{\text{Reverse}} LLRG_{(4)}\ (L^R)^R = L$$

(1) → (A) —a→ (B) ⟲ a, b (2) Reverse ((A)) ←a— (B) ⟲ a, b

(3) RLRG:

$G_1 \left\{ \begin{array}{l} B \to aB \mid bB \mid aA \\ A \to \varepsilon \end{array} \right\}$

(4) Reverse: LLRG:

$G_2 \left\{ \begin{array}{l} B \to Ba \mid Bb \mid Aa \\ A \to \varepsilon \end{array} \right\}$

LLRG:

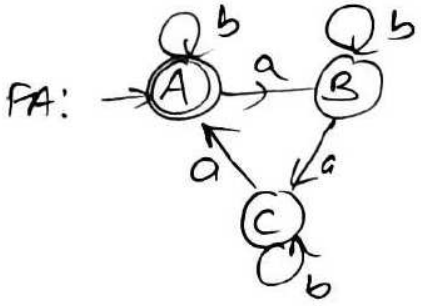
G_1 Ans $\left\{ \begin{array}{l} B \to Ba \mid Bb \mid Aa \\ A \to \varepsilon \end{array} \right\}$

$L(G_1) = a(a+b)^*$, $L_2 = a(a+b)^*$

$L(G_1) = L(G_2)$

Ex:- Convert the FA Given below to Left Linear Grammar.

FA:



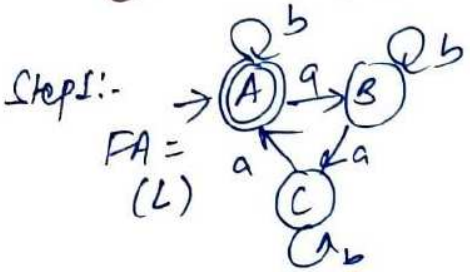
Soln:-

Reverse Reverse

FA → FA → RLRG → LLRG

① (L) ② (L^R) ③ (L^R) ④ $(L^R)^R = L$

Step1:-

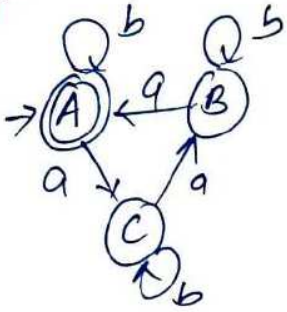


FA = (L)

Reverse OR Transpose Automata:
Interchange initial & Final states
and change Arrow direction

Step 2: Reverse

FA = (L^R)



Step3: RLRG : (L^R)

$$G\begin{cases} A \to aC \mid bA \mid \varepsilon \\ B \to aA \mid bB \\ C \to aB \mid bC \end{cases}$$

Step 4: Reverse

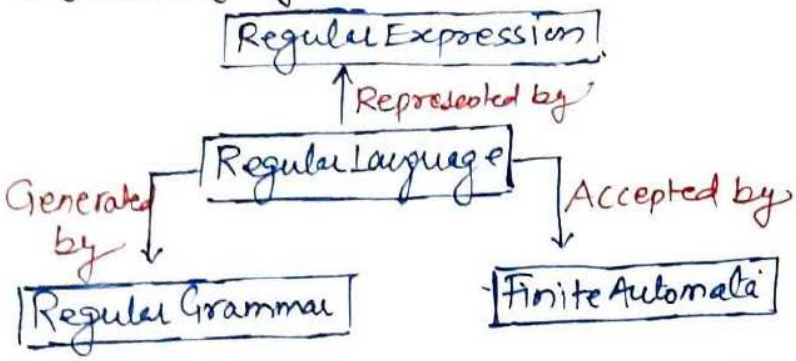
LLRG :

$(L^R)^R = L$

$$G = \begin{cases} A \to Ca \mid Ab \mid \varepsilon \\ B \to Aa \mid Bb \\ C \to Ba \mid Cb \end{cases}$$

Ans:

Regular Language:



- A Language is accepted by FiniteAutomata is called regular Language
- A Language is generated by Regular Grammar is called regular Language
- A Language is represented by Regular expression is called regular Language.

Regular Expression:- Regular expression is a way of describing regular language in Algebraic form.

Rules of R.E: This notation involves a combination of string of symbols from some alphabet Σ, parenthesis, and the operator, $+$, $\cdot$ and $*$, The simplest case in the language $\{a\}$ which will be denoted by the regular expression a

1- **Operators:** $+$, $\cdot$, $*$, $($, $)$ etc.

2- ϕ, ε and $\forall a \in \Sigma$ are primitive regular expression

3- if r_1 and r_2 are regular expression so are r_1+r_2, $r_1 \cdot r_2$, r_1^*, r_2^*. (r_1), (r_2), (r_1+r_2), $(r_1 \cdot r_2)$, $(r_1+r_2)^*$, $(r_1 \cdot r_2)^*$ regular expression.

4- A string is a regular expression iff it can be derived from the Primitive regular expression

Ex:- Regular language	Regular expression:
$L = \{\}$	$R.E = \phi$
$L = \{\varepsilon\}$	$R.E = \varepsilon$
$L = \{a\}$	$R.E = a$
$L = \{a, b, c\}$	$RE = a+b+c$
$L = \{\varepsilon, a, aa, aaa \ldots\}$	$RE = a^*$
$L = \{a, aa, aaa, aaaa \ldots\}$	$RE = a+$
$L = \{\varepsilon, a, b, aa, ab, ba, bb \ldots\}$	$RE = (a+b)^*$

0 0

Precedance of operators:

( )
(*/+)
•
∩
∪ or +

Decrease ↓

Ex: $\phi \cap \varepsilon \cup a^* = ?$

$\phi \cap \varepsilon \cup a^*$

$((\phi \cap \varepsilon) \cup a^*)$ $(\phi \cap (\varepsilon \cup a^*))$

$\phi \cup a^*$ $\phi \cap a^*$

Ans: a^* ✓ Correct ϕ Wrong

Associativity: Left to Right

Ex: $L_1 = \phi$, $L_2 = \{a\}$, Evaluate $L_1 \cdot L_2^* \cup L_1^* = ?$

$$L_1 \cdot L_2^* \cup L_1^* = \phi \cdot a^* \cup \phi^* = \phi \cup \varepsilon = \varepsilon \text{ Ans.}$$

Write Regular expression for the following Languages.

- $L_1 = \{w \mid w \in (a,b)^*\}$, $\Sigma = \{a,b\}$
  - i) $|w| = 2 \Rightarrow$ R.E = $(a+b)(a+b)$
  - ii) $|w| \geq 2 \Rightarrow$ R.E = $(a+b)(a+b)(a+b)^*$
  - iii) $|w| \leq 2 \Rightarrow$ RE = $(a+b)(a+b) + (a+b) + \varepsilon = (a+b+\varepsilon)(a+b+\varepsilon)$
  - iv) $|w|$ = Even $\Rightarrow$ RE = $[(a+b)(a+b)]^*$
  - v) $|w|$ = odd $\Rightarrow$ RE = $[(a+b)(a+b)]^*(a+b)$

$L_2 = \{w \mid |w| \bmod 3 = 0, w \in (a,b)^*\} \Rightarrow$ R.E = $[(a+b)(a+b)(a+b)]^*$

$\neq 0 \Rightarrow$ RE = $[(a+b)(a+b)(a+b)]^*[(a+b)+(a+b)(a+b)]$

L_3 = Set of all strings start with a, over $\Sigma = \{a,b\} \Rightarrow$ RE = $a(a+b)^*$

L_4 = Set of all strings end with a, over $\Sigma = \{a,b\} \Rightarrow$ RE = $(a+b)^*a$

L_5 = Set of all strings Contain a as a Substring $\Rightarrow$ $(a+b)^*a(a+b)^*$

L_6 = Set of all strings end start with a & end with a. $\Rightarrow$ $a(a+b)^*a$

L_7 = Start & end with different symbol $\Rightarrow$ $a(a+b)^*b + b(a+b)^*a$

L_8 = Start & end with same Symbol $\Rightarrow$ $a(a+b)^*a + b(a+b)^*b + a + b + \varepsilon$

L_9 = Set of all strings Contain atleast two a $\Rightarrow$ $(a+b)^*a(a+b)^*a(a+b)^*$

L_{10} = Contain exact two a $\Rightarrow$ $b^*ab^*ab^*$

L_{11} = Contain atmost two a $\Rightarrow$ $b^*ab^*ab^* + b^*ab^* + b^*$

L_{12} = set of all strings contain aa as a substrings over $\Sigma = \{a,b\}$

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

L_{13} = Set of all strings does not contain aa as a substrings over $\Sigma = \{a,b\}$

$RE = (ab+b)^* + (ab+b)^* a = (ab+b)^* (\varepsilon + a)$ OR $(\varepsilon + a)(b+ba)$

L_{14} = Set of all strings contain even no. of a, over $\Sigma = \{a,b\}$

$RE = (b^* ab^* ab^*)^* + b^*$ OR $b^* (ab^* ab^*)^*$ OR $(b^* ab^* a)^* b^*$

L_{15} = Set of all strings contain atleast one pair of a

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

L_{16} = Every 0 immediately followed by at least two 1, over $\Sigma = \{0,1\}$

$RE = (1 + 011)^*$

$L_{17} = \{ a^m b^n \mid m,n \geq 0 \}$

$RE = a^* b^*$

$L_{18} = \{ a^m b^n \mid m+n = \text{Even}, m,n \geq 0 \}$

$RE = (aa)^* (bb)^* + (aa)^* a (bb)^* b$

$L_{19} = \{ (ab)^n \mid n \geq 0 \}$

$RE = (ab)^*$

$L_{20} = \{ a^{2n} \mid n \geq 0 \}$

$RE = (aa)^*$

$L_{21} = \{ 0, 1, 10, 100, 1000, \ldots \} \Rightarrow RE = (0 + 10)^*$

$RE = (0 + 10^*)$

Algebraic Properties of regular expression:

Kleen closure is an unary operator and Union (+) and concatenation operator (•) are binary operators.

1- Closure :- if r_1 and r_2 are regular expression, Then

- r_1^* is a R.E
- $r_1 + r_2$ is a R.E
- $r_1 . r_2$ is a R.E

2. Closure laws:-

- $(r^*)^* = r$, closing an expression that is already closed does not change the Language
- $\phi^* = \varepsilon$, a string formed by concatenating any number of copies of an empty string is empty itself
- $r^+ = r . r^* = r^* . r$, as $r^* = \varepsilon + r + rr + rrr \ldots$ and $r . r^* = r + rr + rrr \ldots$
- $r^* = r^+ + \varepsilon$

3. Associativity:- if r_1, r_2, r_3 are R.E, Then

i) $r_1 + (r_2 + r_3) = (r_1 + r_2) + r_3$

- For Ex:- $r_1 = a$, $r_2 = b$, $r_3 = c$, Then
  - The resultant regular expression in LHS becomes $a + (b + c)$ and the regular set for the corresponding RE is $\{a, b, c\}$
  - for the RE in RHS becomes $(a + b) + c$ and the regular set for this RE is $\{a, b, c\}$ which is same in both cases, Therefore, The associativity property holds for Union operator

Associativity Property does not hold for Kleen closure (*) because it is unary operator

4 - Identity:- In the case of Union operators

if $r + x = r \Rightarrow x = \phi$ as $r \cup \phi = r$, Therefore ϕ is the identity for +.

Therefore, ϕ is the identity element for a Union operator.

In the case of Concatenation operator -

if $r . x = r$, for $x = \varepsilon$

$r . \varepsilon = r \Rightarrow \varepsilon$ is the identity element for Concatenation operator (•).

5- Annihilator :-

- if $r + x = r \Rightarrow r \cup x = x$, There is no annihilator for +
- in the case of concatenation operator, $r \cdot x = x$, when $x = \phi$, Then $r \cdot \phi = \phi$, Therefore ϕ is the annihilator for the ($\cdot$) operator

For EX:- $\{a, aa, ab\} \cdot \{\} = \{\}$

6- Commutative Property :- if r_1, r_2 are R.E, then

- $r_1 + r_2 = r_2 + r_1$. for Example, for $r_1 = a$ and $r_2 = b$ Then R.E $a+b$ and $b+a$ are equal
- $r_1 \cdot r_2 \neq r_2 \cdot r_1$. for EX:- for $r_1 = a$ and $r_2 = b$, Then RE. $a.b$ is not equal to $b.a$

7. Distributed Property :- if r_1, r_2, r_3 are regular expressions, then

- $(r_1 + r_2) \cdot r_3 = r_1 \cdot r_3 + r_2 \cdot r_3$ i.e Right distribution
- $r_1 \cdot (r_2 + r_3) = r_1 \cdot r_2 + r_1 \cdot r_3$ i.e left distribution
- $(r_1 \cdot r_2) + r_3 \neq (r_1 + r_3)(r_2 + r_3)$

8- Idempotent law :-

- $r_1 + r_1 = r_1 \Rightarrow r_1 \cup r_1 = r_1$, Therefore the Union operator Sahsfies idempotest property
- $r \cdot r \neq r \Rightarrow$ Concatenation operator does not Salisfy idempotest property.

Identities Related to Regular Expression:-

There are many identities for the regular expression. Let P, Q and r are regular expressions.

1. $\phi + R = R$
2. $\phi \cdot R = R\phi = \phi$
3. $\varepsilon \cdot R = R \cdot \varepsilon = R$
4. $\varepsilon^* = \varepsilon$, $\phi^* = \varepsilon$
5. $R + R = R$
6. $R^* \cdot R^* = R^*$
7. $R \cdot R^* = R^* \cdot R = R^+$
8. $(R^*)^* = R^*$
9. $\varepsilon + RR^* = \varepsilon + R^*R = R^*$
10. $(PQ)^*P = P(QP)^*$
11. $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^* = P^*(QP^*)^*$ (Imp)
12. $(P+Q)R = PR + QR$ and $R(P+Q) = RP + RQ$

NOTE:- Two regular expression P and Q are equivalent ($P = Q$) if P and Q Represent the same set of Strings.

Ex:- Simplify regular expression $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$

Soln:
$$= (1+00^*1)[\varepsilon + (0+10^*1)^*(0+10^*1)]$$
$$\because \varepsilon + R^*R = R^*$$
$$= (1+00^*1)(0+10^*1)^*$$
$$= (\varepsilon + 00^*)1(0+10^*1)^* \quad \because \varepsilon + RR^* = R^*$$
$$= 0^*1(0+10^*1)^*$$ Ans

Ex: the regular expression $0^*(10^*)^*$ denotes the same set as

a) $(1^*0)^*1^*$ — Soln: $0^*(10^*)^* = (0+1)^*$ ∵ identity No (11)

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

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

d) None.

$(1^*0)^*1^* = 1^*(01^*)^*$ by identity (10)

Now $1^*(01^*)^* = (1+0)^*$ by identity (11)

Hence $(0+1)^* = (1+0)^*$ proved. Ans (a)

Ex:- Let $A = (m+n^*)^*$, $B = (m+n)^*$.

TRUE:
(a) $A = B$ (b) $A \subset B$ (c) $B \subset A$ (d) $A \cap B = \phi$

Soln: $A = (m+n^*)^* = (m^*(n^*)^*)^* = (m^*n^*)^* = (m+n)^* = B$

By identity (8)

Ex:- $A = [(01)^*1^*]$

$B = (01+1)^*$ TRUE: (a) $A = B$ (b) $B \subset A$ (c) $A \subset B$ (d) None.

Soln: $B = (01+1)^* \ldots [(01)^*1^*] \Rightarrow A \subset B$ Ans

Ex:- Write a regular expression for the set of strings of 0's and 1's not containing 101 as a substring.

Soln:

$$RE = 0^*(1^*000^*)^*1^*0^*+(010)^*$$

Ex:- Write the regular expression for the set of strings of 0's and 1's whose length symbol from the right end is 1.

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

Ex: $L = \{a^{2n}b^{2m+1} \mid n \geq 0, m \geq 0\}$ R.E = ?

Ans: $RE = (aa)^*(bb)^*b$

Ex: Write regular expression for $L = \{a^n b^m \mid n \geq 4, m \leq 3\}$

Ans: $R.E = a^4a^*(\varepsilon+b+b^2+b^3)$

Ex:- $L = \{w \mid |w| \bmod 3 = 0\}, w \in (a,b)^*$

Ans: $RE = [(a+b)(a+b)(a+b)]^*$

Ex:- $L = \{w \mid w \in (a,b)^*, n_a(w) \bmod 3 = 0\}$

Soln: $RE = (b^*ab^*ab^*ab^*)^*+b^*$

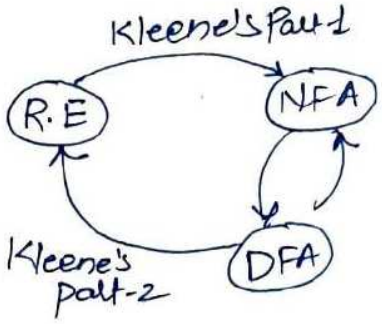
Ex:- Write the regular expression for the language containing the strings over $\{0,1\}$ in which there are atleast two occurrence of 1's between any two occurrence of 0's

Soln: $RE = (0(11)^*0(11)^*0)^*$

Ex:- Write the regular expression for the language; The set of all string in which every pair of adjacent zero's appear before any pair of adjacent one's

Soln: $(0011)^*$

Kleene's Theorem :-



Kleene's Part1 :-

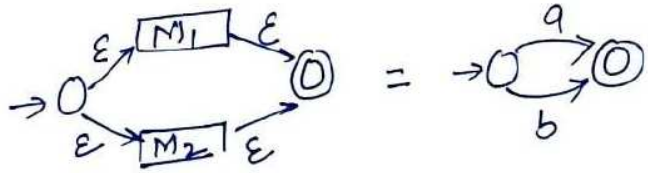
R.E to NFA :

1- ϕ = 

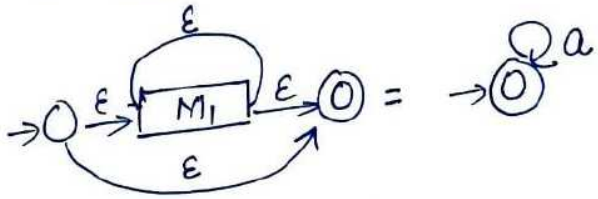
2- ε = 

3- a =  = M_1

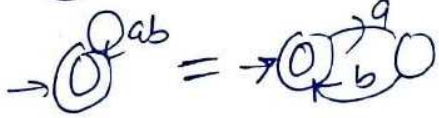
4- b =  = M_2

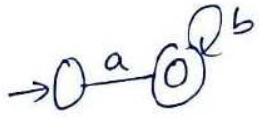
5- a+b = 

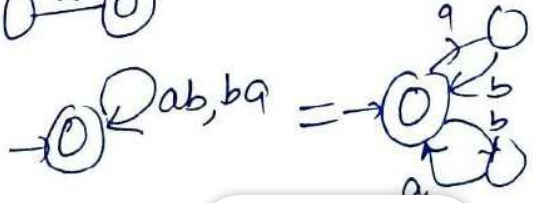
6- a.b = 

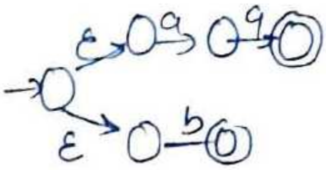
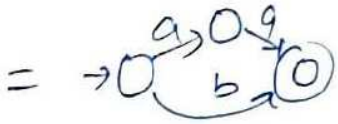
7- a^* = 

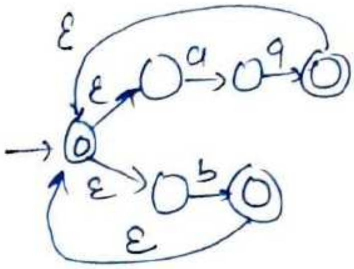
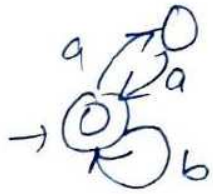
8- $(a+b)^*$ = 

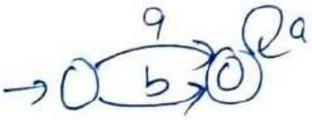
9- $(ab)^*$ = 

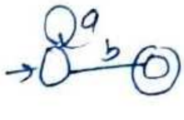
10- ab^* = 

11- $(ab+ba)^*$ = 

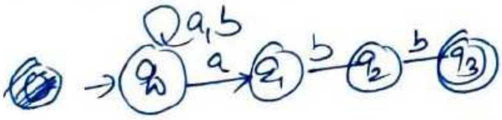
12. $aa+b$ =  = 

13. $(aa+b)^*$ =  = 

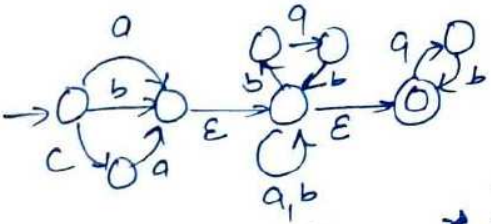
14. $(a+b)a^*$ = 

15. a^*b = 

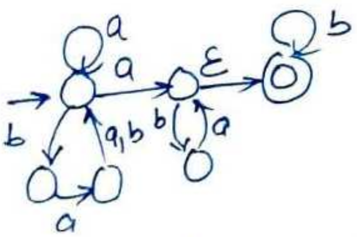
16 RE = $(a+b)^*abb$, FA = ?

FA = 

17. RE = $(a+b+ca)(bab+(a+b))^*(ab)^*$

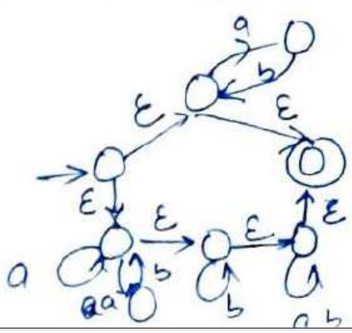


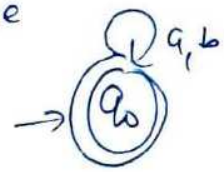
18. RE = $[a+ba(a+b)]^*a(ba)^*b^*$

FA:  Ans:

19. RE = $(ab)^* + (a+ab)^* b^*(a+b)^*$

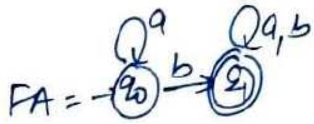
⟹ minimize $(a+b)^*$



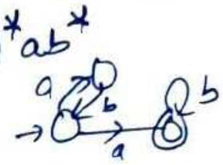
OR 

... m State = 2 Ans

20. RE = $a^*b(a+b)^*$

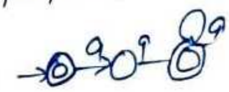
FA = 

21. $(ab)^*ab^*$

FA = 

22. $(aa+aaa)^*$

L = { ε, aa, aaa, aaaa, ... }

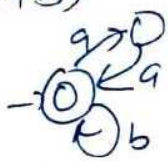
FA = 

23. $(a+aaaa)^* = a^*$

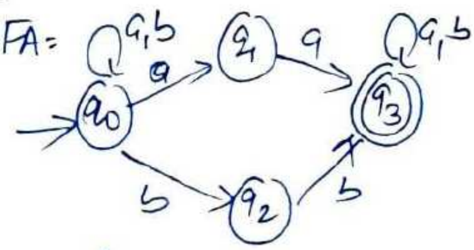
L = { ε, a, a^2, a^3, a^4 ... }

FA = 

24. $(aa+b)^*$

FA = 

25. $(a+b)^*(aa+bb)(a+b)^*$

FA: 

Ans.

Conversion of R.E to FA: To convert the R.E to FA, we are going to use a method called the subset method. This method is used to obtain FA from the given regular expression. This method is given below.

Step1: Design a transition diagram for given regular expression, using NFA with ε-moves

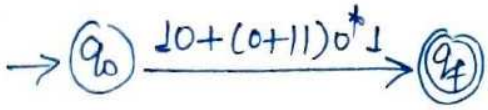
Step2: Convert this NFA with ε to NFA without ε.

Step3: Convert the obtained NFA to equivalent DFA.

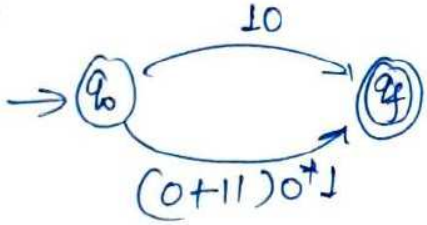
Example: Design a FA from given regular expression $10+(0+11)0^*1$

Sol^n: First we will construct the transition diagram for a given regular expression

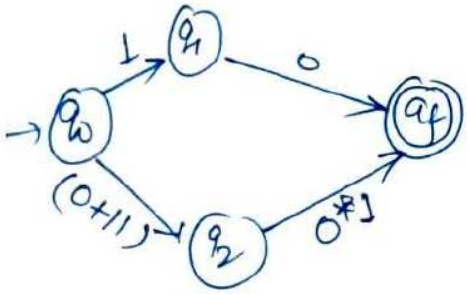
Step1:



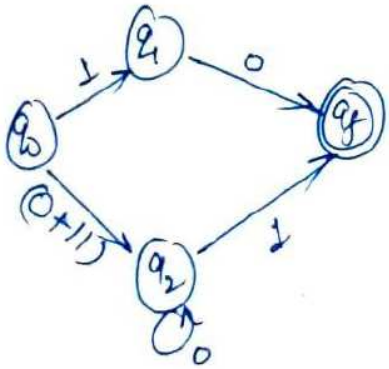
Step2:



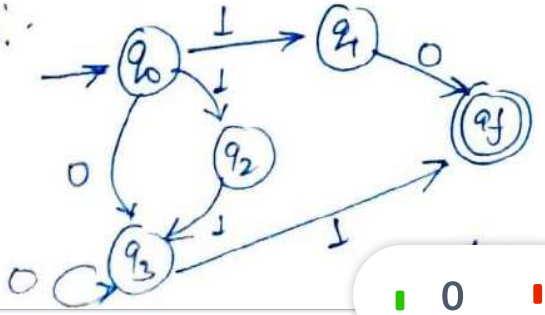
Step3:



Step4:-



Step5:

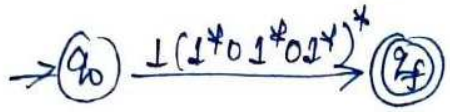


Now we have got NFA without ε. Now we will convert it into required DFA for that, ~~we will first write a transition table for this NFA~~

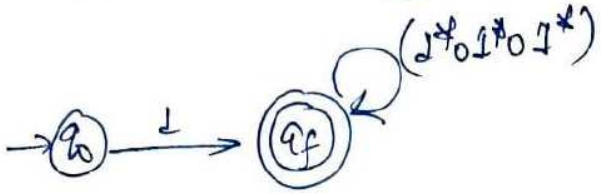
Ex:- Design a NFA from given regular expression $1(1^*01^*01^*)^*$

Sol^n:. The NFA for the given regular expression is as follows

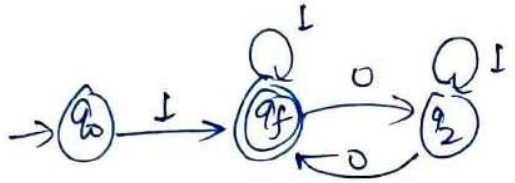
Step1:-



Step2:-



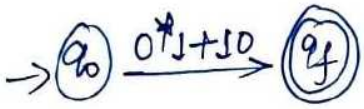
Step3:-



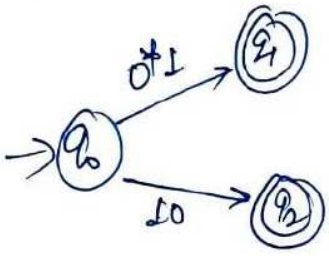
Ex:. Construct the FA for regular expression 0^*1+10

Sol^n.

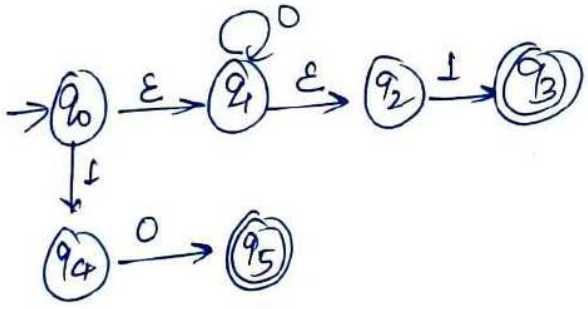
Step1:



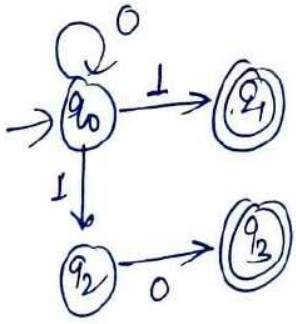
Step2:



Step3:



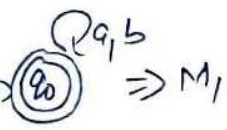
Step4:-

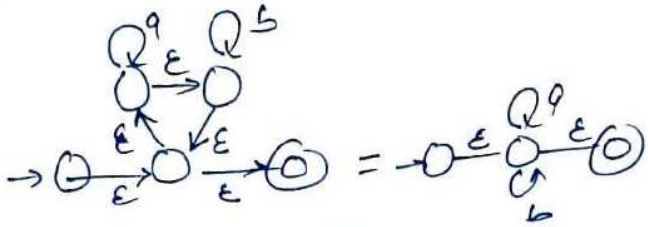


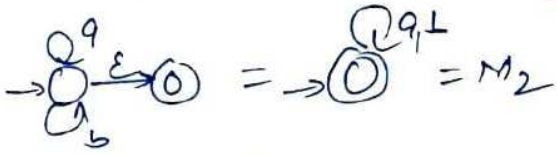
Ans.

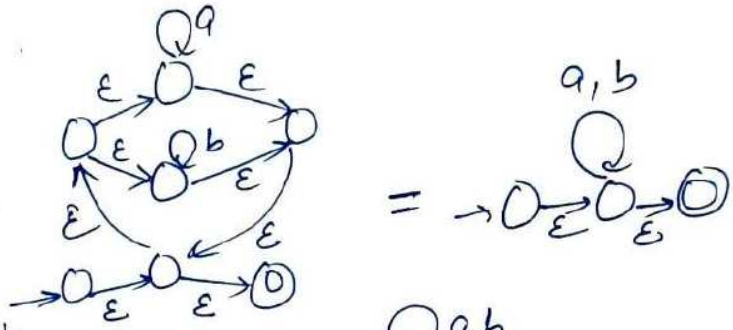
Proof: $(a+b)^* = (a^*.b^*)^* = (a^*+b^*)^* = a^*(ba^*)^*$

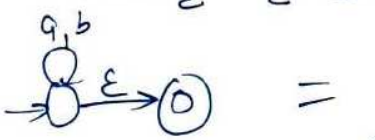
Sol^n:

1- $(a+b)^* =$  $\Rightarrow M_1$

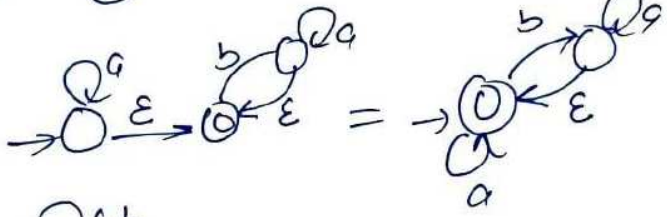
2- $(a^*.b^*)^* =$ 

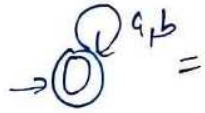
$=$  $= M_2$

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

$=$  $= M_3$

4- $a^*(ba^*)^*$

$=$ 

$=$  $= M_4$

$\therefore$ Now $M_1 = M_2 = M_3 = M_4 =$ 

Hence Proved: $\boxed{(a+b)^* = (a^*b^*)^* = (a^*+b^*)^* = a^*(ba^*)^*}$

Ex:- Let r_1 and r_2 be two regular expressions defined as follows:
$r_1 = (00^*1)^*1$ and $r_2 = 1+0(0+10)^*11$, Prove that $r_1 = r_2$

Sol^n:- Two regular expression P and Q are equivalent iff P and Q are represents the same set of strings

$r_1 = (00^*1)^*1 = \{1, 011, 0011, 00011, 01010101, \cdots\}$

$r_2 = 1+0(0+10)^*11 = \{1, 011, 0011, 00011, 01010101, \cdots\}$

Since both regular expressions Produce same set of string

$\therefore r_1 =$

Finite Automata to regular expression:

Arden's Theorem: Let P, Q and R be three regular expression if P does not contain ε, Then $R = Q + RP$ has a unique solution that is $\boxed{R = QP^*}$, if P contain ε then $R = Q + RP$ has more than one solution.

That means, whenever we get any equation in the form of $R = Q + RP$ Then we can directly replaced by $R = QP^*$.

Assumption for applying Arden's Theorem:

1. The transition diagram must not have ε-transition
2. It must have only one initial state
3. Arden's Theorem used only for DFA, NFA but not for ε-NFA

State and prove Arden's Theorem:

$$R = Q + RP \quad \ldots (i)$$

Now, replacing R by $R = QP^*$, we get

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

$$R = Q(\varepsilon + P^*P) \qquad \because \varepsilon + R^*R = R^*$$

$$\boxed{R = QP^*}$$ Hence proved.

Thus, $R = QP^*$ is the solution of the equation $R = Q + RP$, Now we have to prove that this is the only solution of this equation. Let me take this equation again $R = Q + RP$, Now replace $R = Q + RP$

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

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

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

$$R = Q + QP + QP^2 + RP^3 \text{ again replace } R = Q + RP$$

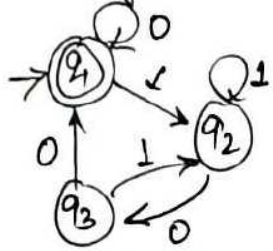
$$R = Q + QP + QP^2 + QP^3 + RP^4, \text{ again replace } R = Q + RP$$

$$R = Q + QP + QP^2 + QP^3 + QP^4 + \ldots\ldots$$

$$R = Q(\varepsilon + P + P^2 + P^3 + P^4 + \ldots\ldots)$$

$$\boxed{R = QP^*}$$, Hence proved.

Ex:. Find regular Expression



Sol^n:. First find equation of each state; After that find regular expression for final state

$q_1 = q_1 \cdot 0 + q_3 \cdot 0 + \varepsilon$ ----(i)

$q_2 = q_1 \cdot 1 + q_2 \cdot 1 + q_3 \cdot 1$ ----(ii)

$q_3 = q_2 \cdot 0$ ----(iii)

If More than one state final Then add all final state regular expression

Now $q_2 = q_1 \cdot 1 + q_2 \cdot 1 + q_2 \cdot 0 \cdot 1$

$q_2 = q_1 \cdot 1 + q_2 (1 + 01)$

Arden's: $R = Q + RP \Rightarrow QP^*$

$q_2 = q_1 \cdot 1 (1+01)^*$, Now Put q_2 value in equation (iii)

$q_3 = q_1 \cdot 1 (1+01)^* 0$, Now Put q_3 value in equation (i)

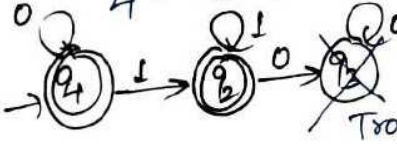
$q_1 = q_1 \cdot 0 + q_1 \cdot 1 (1+01)^* 0 \cdot 0 + \varepsilon$

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

Arden's $R = Q + RP \Rightarrow QP^*$

$q_1 = \varepsilon \cdot (0 + 1 (1+01)^* 00)^* = (0 + 1 (1+01)^* 00)^*$ Ans

Ex:.



Find regular expression = ?

Sol^n: $q_1 = q_1 \cdot 0 + \varepsilon$ Remove Trap or Dead state

$q_2 = q_1 \cdot 1 + q_2 \cdot 1$

$q_1 = q_1 \cdot 0 + \varepsilon$ --(i)

$q_2 = q_1 \cdot 1 + q_2 \cdot 1$ --(ii)

Now $q_1 = \varepsilon + q_1 \cdot 0$

By Ardens $q_1 = \varepsilon \cdot 0^* = 0^*$, put q_1 value in equation (ii)

$q_2 = 0^* \cdot 1 + q_2 \cdot 1$, $\because R = Q + RP \Rightarrow QP^*$

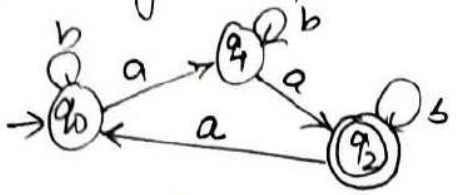
$q_2 = 0^* 1 1^*$

Final Ans $\Rightarrow q_1 + q_2 = 0^* + 0^* 1 1^*$

$= 0^* (\varepsilon + 1 1^*)$ $\because \varepsilon + RR^* = R^*$

$= 0^* 1^*$ Ans

Ex:. Find the regular expression of given FA using Arden's theorem [AKTU 2018-19, Marks 10]



Sol^n:

$q_0 = q_0 \cdot b + q_2 \cdot a + \varepsilon$ ---- (i)

$q_1 = q_0 \cdot a + q_1 \cdot b$ ---- (ii)

$q_2 = q_1 \cdot a + q_2 \cdot b$ -- (iii)

Arden's: $R = \underbrace{Q}_{Q} + \underbrace{R}_{R}\underbrace{P}_{P} \Rightarrow QP^*$

$q_2 = q_1 \cdot a\, b^*$

Now $q_1 = q_0 \cdot a + q_1 \cdot b$

Arden's: $R = Q + RP \Rightarrow QP^*$

$q_1 = q_0 \cdot a\, b^*$

Now: $q_2 = q_0\, ab^* \cdot ab^*$, put q_2 value in equation (i)

Now $q_0 = \varepsilon + q_0 \cdot b + q_0 \cdot ab^* \cdot ab^* a$

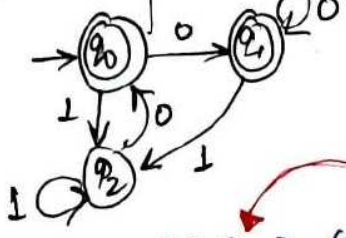
$q_0 = \varepsilon + q_0 (b + ab^* ab^* a)$

Arden's: $R = Q + RP \Rightarrow QP^*$

$q_0 = \varepsilon \cdot (b + ab^* ab^* a)^* = (b + ab^* ab^* a)^*$

Now Ans $\boxed{q_2 = (b + ab^* ab^* a)^* ab^* \cdot ab^*}$ Ans.

Ex:. Find regular expression of given FA, using Arden's Theorem [AKTU-2017-18, Marks-07]



Sol^n:

$q_0 = \varepsilon + q_2 \cdot 0$ -- (i)

$q_1 = q_0 \cdot 0 + q_1 \cdot 0$ -- (ii)

$q_2 = q_0 \cdot 1 + q_1 \cdot 1 + q_2 \cdot 1$ --- (iii)

put value of q_0 and q_1 in equation (iii)

$q_2 = q_0 \cdot 1 + (q_0 \cdot 00^*) 1 + q_2 \cdot 1$

Now Apply Arden's for equation (ii)

$q_1 = q_0 \cdot 00^*$

$q_2 = q_0 \cdot 1 + q_0 \cdot 00^* 1 + q_2 \cdot 1$

$q_2 = q_0 (1 + 00^* 1) + q_2 \cdot 1$

$q_2 = q_0 (1 + 00^* 1) 1^*$ ∵ By Arden's

$q_0 = \varepsilon + q_0 (1 + 00^* 1) 1^* 0$

$q_0 = [(1 + 00^* 1) 1^* 0]^*$ By Arden's

$q_2 = [(1 + 00^* 1) 1^* 0]^* [1 + 00^* 1] 1^*$

Final Ans $q_1 = [(1 + 00^* 1) 1^* 0]^* 00^*$

L.Q. $= [(1 + 00^* 1) 1^* 0]^* [\varepsilon + 00^*]$ Ans

FA to R.E Direct Method:

1 → 

$= (x+y+z)^*$

2 → 

loop at A: loop at B

RE ⇒ $(PQ)^*P = P(QP)^*$

3 → 

loop at A:

$A = (PQ)^*$

$B = (PQ)^*P$

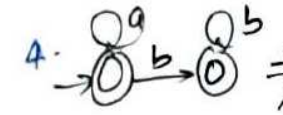
$Ans = A+B = (PQ)^*(\varepsilon+P)$

loop at B

$A = P(QP)^*Q+\varepsilon$

$B = P(QP^*)$

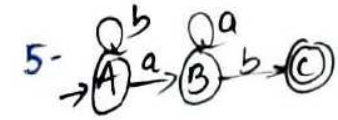
$Ans = A+B = P(QP)^*Q+P(QP)^*+\varepsilon$

4. 

⇒ A: a^*

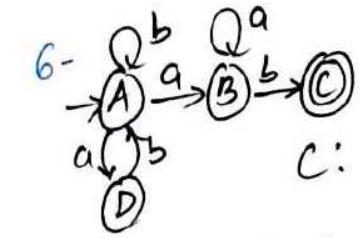
B: a^*bb^*

$RE = A+B = a^*(\varepsilon+bb^*) = a^*b^*$ Ans

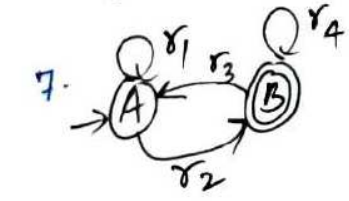
5- 

, RE for C: b^*aa^*b Ans

b/c C is final state.

6- 

C: $(b+ab)^*aa^*b$

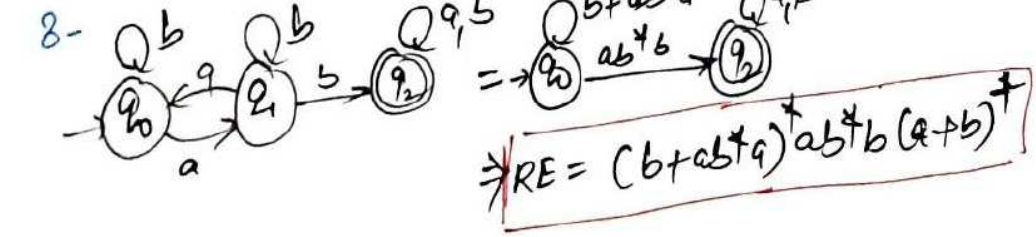
7. 

loop at A:

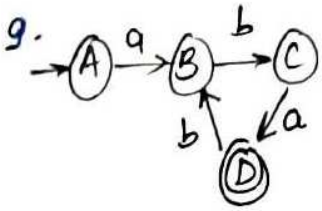
$RE = (r_1+r_2r_4^*r_3)^*r_2r_4^*$

loop at B:

$R.E = r_1^*r_2(r_4+r_3r_1^*r_2)^*$

8- 

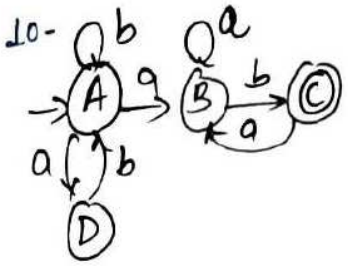
$\Rightarrow RE = (b+ab^*a)^*ab^*b(a+b)^*$

9. 

loop at B: $RE = a(bab)^* ba$

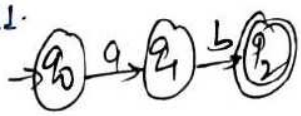
loop at C: $RE = ab(abb)^* a$

loop at D: $RE = aba(bba)^*$

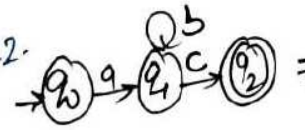
10. 

loop at B:
$RE = (b+ab)^* a(a+ba)^* b$

loop at C:
$RE = (b+ab)^* aa^* b(aa^* b)^*$

11. 

$\Rightarrow ab$

12. 

$\Rightarrow ab^*c$

13. 

$\Rightarrow (ab)^*$

14. 

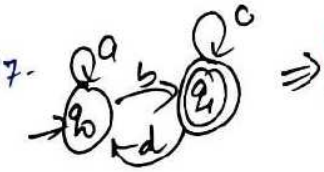
$\Rightarrow a(ba)^*$

15. 

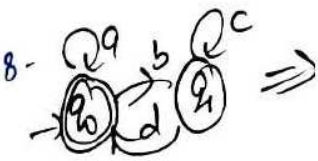
$\Rightarrow (a+bc)^*$

16. 

$\Rightarrow a(b+ca)^*$

17. 

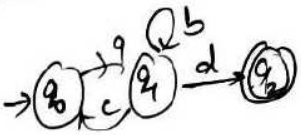
$\Rightarrow a^* b(c+da^* b)^*$

18. 

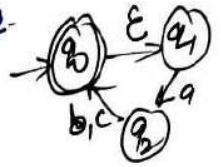
$\Rightarrow (a+bc^* d)^*$

19. 

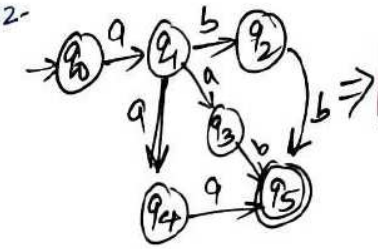
$\Rightarrow a(ba)^* c$

20. 

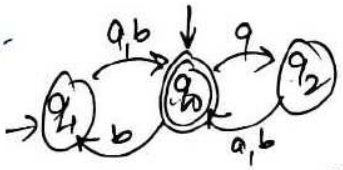
$\Rightarrow a(b+ca)^* d$

21. 

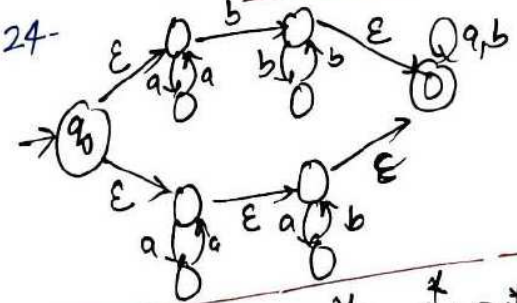
$\Rightarrow [\varepsilon \cdot a(b+c)]^*$
$\Rightarrow [a(b+c)]^*$

22. 

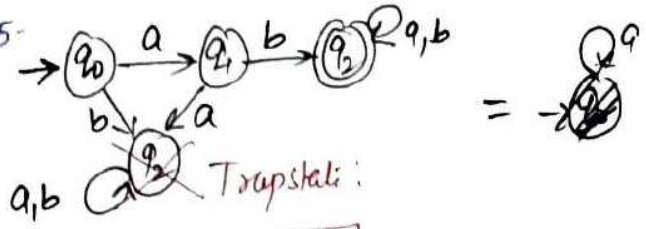
$\Rightarrow a(ba+bb+ab)$

23. 

$= [b(a+b) + a(a+b)]^*$
$= [(a+b)(a+b)]^*$

24. 

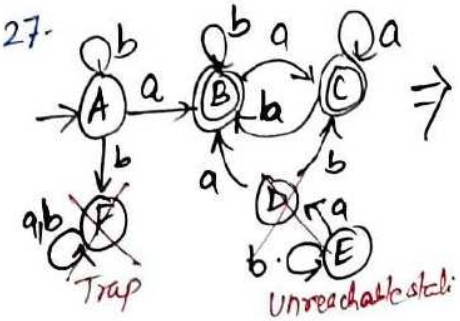
$RE = (aa)^* b(bb)^*(a+b)^* + (aa)^*(ab)^*(a+b)^*$

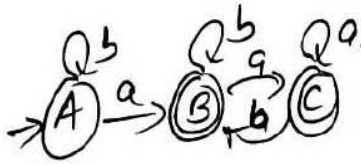
25- 

$\Rightarrow \boxed{ab(a+b)^*}$

26. 

$\Rightarrow \boxed{\phi}$

27- 

$\Rightarrow$ 

Loop at B:- B: $b^*a(b+aa^*b)^*$

C: B: aa^*

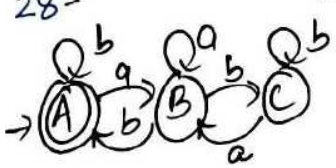
C: $b^*a(b+aa^*b)^*aa^*$

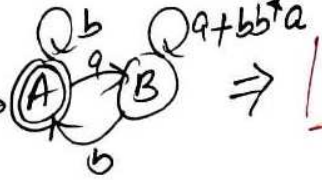
Final Ans $\Rightarrow B+C = b^*a(b+aa^*b)^*(\varepsilon+aa^*)$

$= b^*a((\varepsilon+aa^*)b)^*a^*$

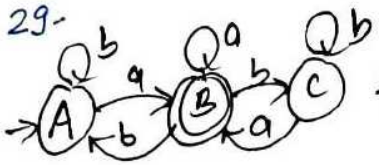
$= b^*a(a^*b)^*a^*$

$= b^*a\,a^*(ba^*)^* \Rightarrow \boxed{b^*a(a+b)^*}$ Ans $=$ 

28- 

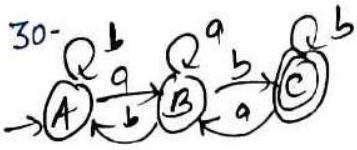
$\Rightarrow$ 

$\Rightarrow \boxed{[b+a(a+bb^*a)^*b]^*}$

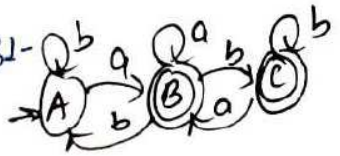
29- 

$\Rightarrow b^*a(a+bb^*a)^*$

$\Rightarrow \boxed{b^*a(b^*a)^*}$

30- 

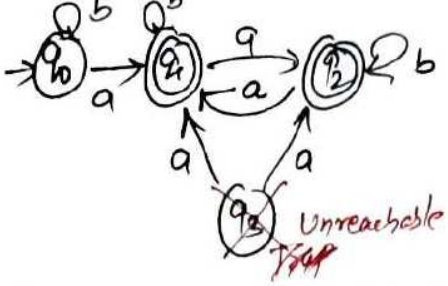
$\Rightarrow b^*a(a+bb^*a+bb^*a)^*bb^* = \boxed{b^*a(a+bb^*a)^*bb^*}$

31- 

$\Rightarrow B+C = b^*a(a+bb^*a)^*(\varepsilon+bb^*)$

$= \boxed{b^*a(b^*a)^*b^*}$

B2:-



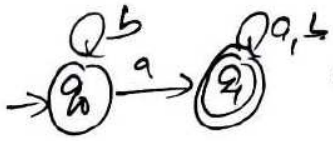
$RE = b^*a(a+b)^*$

Q-1:. The Language accepted by this automata is given by R.E.

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

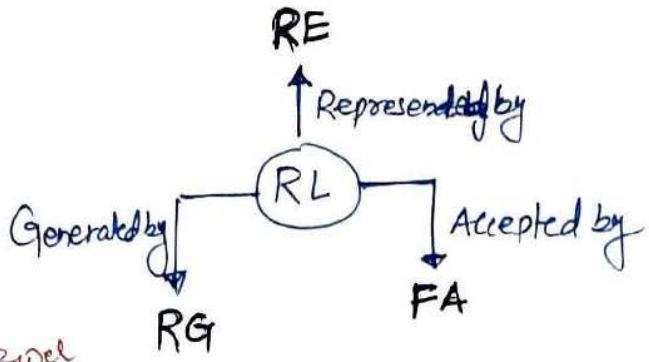
Q-2:. The minimum state automata equivalent to the above FSA has the following No. of states

a) 1 b) 2 c) 3 d) 4



ministate = 2.

Regular Language (RL):



Imp:
Every finite language is regular language

Power

$$P(RG) \equiv P(RL) \equiv P(RE) \equiv P(FA)$$

Ex: Regular language Examples:

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

2- $L = \{a^{xn} \mid x$ is any Positive integer $n \geq 0\}$

3- $L = \{a^m b^n \mid m, n \geq 0\}$

4- $L = \{a^m b^n c^k \mid m, n, k \geq 0\}$

5- $L = \{a^n b^n c^n \mid 1 \leq n \leq 10^5\}$

6- $L = \{a^m b^n,\ m+n$ is even No. $m, n \geq 0\}$

7- $L = \{a^m b^n \mid m+n = 10, \neq 10, \geq 10, \leq 10, > 10, < 10\}$ Regular

8- $L = \{a^m b^n \mid m*n = 10, \neq 10, \geq 10, \leq 10, > 10, < 10\}$ Regular

9- $L = \{w_1 . w_2 \mid w_1, w_2 \in (a,b)^*\}$

10- $L = \{w x w^R \mid w \in (a,b)^*, x \in (a,b)^+\}$

11- $L = \{w x w y \mid w, x, y \in (a,b)^+$

RE = $a(a+b)^+ a(a+b)^+ + b$ ...

12- $L = \{x w y w \mid w, x, y \in (a,b)^+$

RE = $(a+b)^+ a (a+b)^+ a + b(a+b)^+ b(a+b)^+$ ...

0 0

Properties of Regular Language:

1- Closure Properties:

2- Decision Properties:

Closure Properties: A closure Property of a language class says that given Languages in the class, an operator (i.e, Union) produces another language in the same class.

Union: $L_1 = R.L,\ L_2 = R.L \Rightarrow L_1 \cup L_2 = R.L$ where r_1, r_2 = Regular expression

Proof: $L_1 = R.L,\ L_2 = R.L \Rightarrow L_1 \cup L_2 = R.L$

$r_1 + r_2 \Rightarrow$ Regular Expression $\Rightarrow L_1 \cup L_2 = R.L$

i.e Regular language is closed Under Union operation

Concatenation: $L_1 = R.L,\ L_2 = R.L \Rightarrow L_1 \cdot L_2 = R.L$ where r_1, r_2 = Regular Expression

Proof: $r_1 \cdot r_2 \Rightarrow$ Regular Expression $\Rightarrow$ Regular language $\Rightarrow L_1 \cdot L_2$.

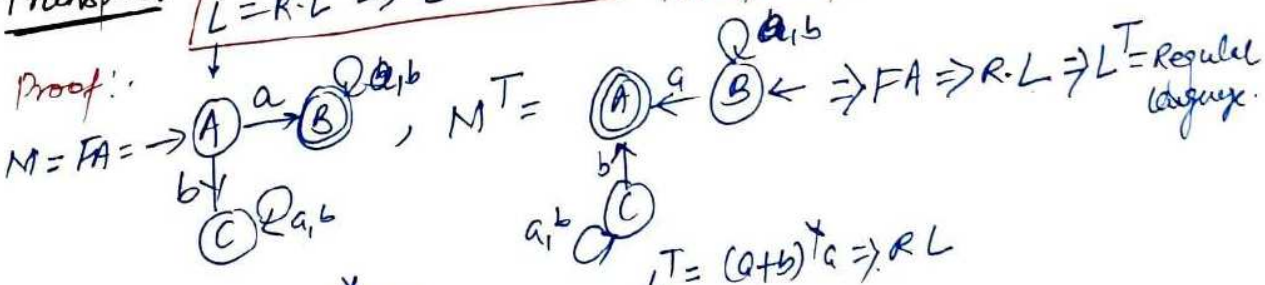
i.e Regular Language is closed Under Concatenation operation

Kleenstar: $L = R.L \Rightarrow L^* = R.L$ i.e Regular language is closed Under Kleen star operation

Proof: $r \Rightarrow r^* = R.E = R.L$

Transpose: $L = R.L \Rightarrow L^T = R.L$ i.e Regular language is closed Under Transpose operation

Proof:

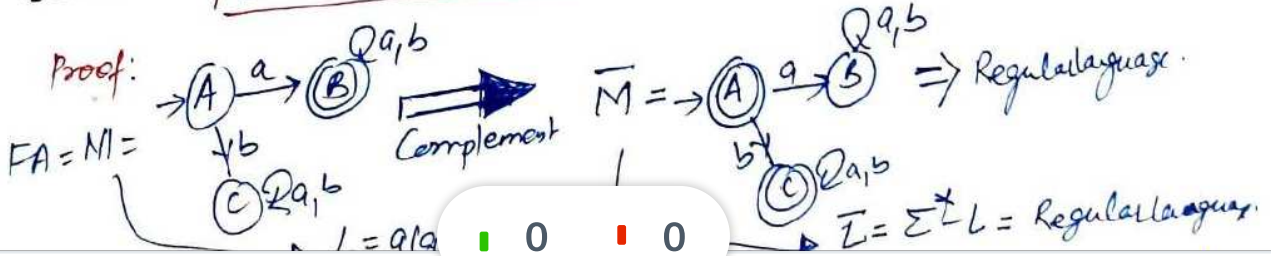


$L = a(a+b)^* = R.L$

$L^T = (a+b)^* a \Rightarrow RL$

Complement: $L = RL \Rightarrow \overline{L} = R.L$ i.e Regular language is closed Under Complement operation

Proof:



$L = a(a+b)$

Imp:

Intersection: $L_1 = R.L,\ L_2 = R.L \Rightarrow L_1 \cap L_2 = R.L$

i.e Regular Language is closed Under intersection operation.

Proof:

$L_1 \Rightarrow R.L \Rightarrow \overline{L_1} = R.L$

$L_2 \Rightarrow R.L \Rightarrow \overline{L_2} = R.L$

$\overline{L_1} \cup \overline{L_2} = R.L \Rightarrow \overline{\overline{L_1} \cup \overline{L_2}} = R.L$

Apply Demorgan's law

$\overline{\overline{L_1} \cap \overline{L_2}} = R.L \Rightarrow L_1 \cap L_2 = R.L$ proved

Set Difference: $L_1 = RL,\ L_2 = R.L \Rightarrow L_1 - L_2 = R.L$

i.e Regular Languages is closed Under Set difference operator

Proof:

$L_1 \Rightarrow RL,\ L_2 = R.L$

$L_1 - L_2 \Rightarrow L_1 \cap \overline{L_2} \Rightarrow R.L$

because $\overline{L_2} = R.L$ and Intersection of two Regular is regular.

Homomorphism: Suppose Σ and Δ are alphabets Then a function $h: \Sigma \to \Delta^*$ is called a homomorphism in words, a homomorphism is a substitution in which a single letter is replaced with a string

If L is a regular language, and h is a homomorphism on its alphabet, Then $h(L) = \{h(w) \mid w \text{ is in } L\}$ is also a regular language.

Proof:- Let E be a regular expression for L

Apply h to each symbol in E.

Language of resulting R.E is h(L).

Ex:- $h(0) = a,\ h(1) = b^*,\ \Sigma = \{0,1\},\ \Delta = \{a,b\}$

$L = 0^*(0+1)1^* \Rightarrow h(L) = ?$

$h(L) = h[0^*(0+1)1^*] = h[0^*].h(0+1).h(1^*)$

$= h(0^*)[h(0)+h(1)]h(1^*) \Rightarrow a^*.b^*$

Ans

NOTE:

$h(\phi) = \phi$

$h(\varepsilon) = \varepsilon$

$h(x^*) = (h(x))^*$

$h(x_1 \cdot x_2) = h(x_1).h(x_2)$

$h(x_1 + x_2) = h(x_1) + h(x_2)$

Inverse Homomorphism: Let h be a homomorphism and L a language whose alphabet is the output language of h.

$$h^{-1}(L) = \{w \mid h(w) \text{ is in } L\}$$

Regular language is closed under Inverse homomorphism

Proof: Start with a DFA A for L.
Construct a DFA B for $h^{-1}(L)$ with:
- The same set of states
- The same start state
- The same final states
- Input alphabet = The symbols to which homomorphism h applies

Ex:- Let $\Sigma = \{0,1,2\}$, $\Delta = \{a,b\}$ Defined h by

$h(0) = a$, $h(1) = ab$, $h(2) = ba$

$L = a(ba)^*$, $h^{-1}(L) = ?$

$L = a\{\varepsilon, ba, baba, bababa, \cdots\}$

$L = \{a, aba, ababa, abababa \cdots\}$

$h^{-1}(L) = \{0, 10, 110, 1110, \cdots\}$ 1^*0
$\{02, 022, 0222, \cdots\}$ 02^*
$\{102, 1102, \cdots\}$ 1^*02^*

$h^{-1}(L) = 1^*0 + 02^* + 1^*02^* = 1^*02^*$ Ans.

Ex:- Let $\Sigma = \{0,1\}$, $\Delta = \{a,b\}$, $h(0) = aa$, $h(1) = aba$, let

$L = (ab + ba)^*a$, $h^{-1}(L) = ?$

Sol^n: $L = \{\varepsilon, ab, ba, abba, baab \cdots\}a$

$L = \{a, aba, baa, abbaa, baaba, \cdots\}$

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

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

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

Quotient operation: Let L_1 and L_2 be Languages on the same alphabet Then Right quotient of L_1 and L_2 is defined as

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

Ex:- $L_1 = \{01, 001, 101, 0001, 1101\}$, $L_2 = \{01\}$

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

Ex:- $L_1 = 10^*1$, $L_2 = 0^*1$, $L_1/L_2 = ?$

Soln: $L_1 = \{11, 101, 1001, 1000, \cdots\}$, $L_2 = \{1, 01, 001, 0001, \cdots\}$

$L_1/L_2 = \{1, 10, 100, 1000, \cdots\} = 10^*$ Ans

Ex:- $101/11 = \phi$

NOTE:- if L is any regular language over Σ Then

(i) $L/\Sigma^* = \text{prefix}(L)$

(ii) $\Sigma^*/L = \Sigma^*$

(iii) $R.L/R.L = R.L$

(iv) $R.L/NR.L = R.L$

(v) $NRL/R.L = NRL$

Decision Properties: A decision Property for a class of languages is an algorithm that takes a formal description of a language (eg a DFA) and tells whether or not some Property holds.

Approximately all the properties are **decidable** in case of finite automata

(i) Emptiness
(ii) Non Emptiness
(iii) Finiteness
(iv) Infiniteness
(v) Membership
(vi) Equality

These are explained as following below.

Emptiness and Non Emptiness: $L = \phi$ OR $L \neq \phi$

Step1: Select the state that can not be reached from the initial state & delete Them (remove Unreachable state).

Step 2: if the resulting machine contains atleast one final states, so then the finite automata accepts the non-empty language

Step 3: if the resulting machine is free from final state, Then finite automata accepts empty language.

Finiteness and Infiniteness: L = Finite OR L = Infinite

Step1: ~~Select the state that can~~ Remove Unreachable state

step2: Remove Dead states

Step3: if the resulting machine contains loops or cycles then the finite automata accepts infinite language.

Step4: if the resulting machine do not contain loops or cycles then the finite automata accepts infinite language.

Membership: If $L = R.L$, Then $\omega \in L$ or $\omega \notin L$

$L = R.L \Rightarrow$ FA

$\omega \rightarrow$ [FA] → Accept Then $\omega \in L$, i.e ω is a member of L
→ Reject Then $\omega \notin L$, i.e ω not member of L

Equality: $L_1 = L_2$ OR $L_1 \neq L_2$

Two finite state automata M_1 & M_2 is said to be equal iff, They accept the same language. minimize the finite state automata and the minimal DFA will be unique.

$L_1 \Rightarrow$ FA = Minimization $\Rightarrow M_1$
$L_2 \Rightarrow$ FA = Minimization $\Rightarrow M_2$

if $M_1 = M_2$ Then $L_1 = L_2$
Otherwise $L_1 \neq L_2$

General properties of Regular language:

1- Every finite language is regular language.

2. Every infinite language need not be regular

3. Ex!: $a^* = \{\epsilon, a, a^2, a^3 \dots\}$ = Regular

$a^n b^n{}_{n \geq 0} = \{\epsilon, ab, a^2b^2, a^3b^3 \dots\}$ = Not Regular.

5. Every non regular language must be infinite

4- Every subset of regular set need not be regular.

Ex!: $L = \{a^m b^n \mid m, n \geq 0\} = a^*b^*$ = Regular.

$a^n b^n \subseteq a^* b^*$

$a^n b^n$ → not Regular, it is CFL

5- Every finite subset of regular set is regular

6- Infinite Union of regular set is non regular

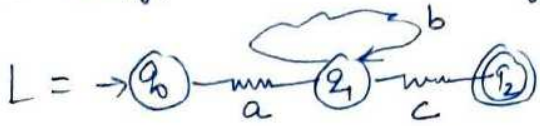
Ex!- $\epsilon \cup ab \cup a^2b^2 \cup a^3b^3 \cup \dots \Rightarrow a^n b^n{}_{n \geq 0}$ ⇒ not Regular.

7- Infinite Intersection of Regular language is non regular.

8- Finite Union and Intersection of Regular language is Regular language.

Pumping Lemma: Pumping Lemma gives a necessary condition it can be used to show that certain sets are not regular

Pumping lemma Theorem: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite Automata with n-states Let L be the regular set accepted by M. Let $w \in L$ and $|w| \geq n$, Then $\exists x, y, z$ such that $w = xyz$, $y \neq \wedge$, $|xy| \leq n$ and $xy^i z \in L$, $\forall i \geq 0$



$L = \{ac, abc, ab^2c, ab^3c, \cdots\}$

Choose $w = abc$, such that ~~$x=a, y=a, z=bc$~~ $x=a, y=b, z=c$, $|xy| \leq n$ and $xy^i z = ab^i c \in L, \forall i \geq 0$

NOTE: ~~$P \to q \equiv \neg q \to \neg P$~~ $P \to q \equiv \neg q \to \neg P$ $\quad$ $P \to q \not\equiv q \to P$

L is regular ⟶ Pumping Lemma hold

¬ Pumping Lemma hold ⟶ ¬ L is regular

NOTE:-

1- Pumping Lemma Satisfy all regular language either finite or infinite

2- All regular language Satisfy Pumping Lemma

3- Non regular language also Satisfy Pumping lemma. means all regular language must Satisfy Pumping lemma but non-regular language may or May not Satisfy Pumping Lemma.

4- if L is regular $\Rightarrow \forall L \exists n \forall w \exists x, y, z \forall xy^i z \in L$

Application of Pumping Lemma: This theorem can be used to Prove that certain sets are not regular.

Step1:- Assume L is regular. Let n be the numbr of states in Corresponding FA

Step2:- Choose a string w such that $|w| \geq n$. used Pumping Lemma to write $w = xyz$, with $|xy| \leq n$ and $|y| > 0$.

Step3: Find a Suitable integer i such that $xy^i z \notin L$ this Contradicts our Assumption hence L is not regular.

0 0

Imp

Ex:- Show that the set $L = \{a^{i^2} \mid i \geq 1$ is not R.L

Soln:

Step1:- Suppose L is regular. Let n be the number of states in FA accepting L

Step2: Let $w = a^{n^2}$, $|w| = n^2$, $|w| > n \Rightarrow n^2 > n$

$w = xyz$, $|xy| \leq n$ and $|y| > 0$

$|w| = |xyz| = |x|+|y|+|z| = n^2$

Step3:- Consider xy^2z

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

$xy^2z \notin L$.

Contradicts Hence L is not regular. Ans

#Imp

Ex:- $L = \{a^p \mid p$ is prime no. is not R.L

Soln:

Step1:- Let L is regular. Let n be the number of states in FA accepting L

Step2:- Let P be a Prime No. greater than n. Let $w = a^p \Rightarrow |w| \geq n$

$\Rightarrow p > n$

$w = xyz$, with $|xy| \leq n$ and $|y| > 0$

So $y = a^m$ for some $m \geq 1$ (and $\leq n$) $\Rightarrow |y| = m$

Let $i = p+1$ Then:

$$|xy^iz| = |xyz| + |y|^{i-1}$$
$$= p + (i-1)|y|$$
$$= p + (i-1)m$$
$$= p + pm$$
$$= p(1+m) \Rightarrow \text{not prime No.}$$

$xy^iz \notin L$

Contradicts Hence L is not regular.

#Imp

Ex: $L = \{a^n b^n \mid n \geq 0\}$, By using Pumping Lemma Show that L is not regular

Sol^n:

Step1: Assume L is a regular Language in order to obtain Contradiction.
Let n be the number of states in finite automata accepting L

Step2: Let $\omega = a^n b^n$, Then $|\omega| = 2n > n$. using Pumping Lemma we can demonstrate ω in three Parts of xyz such that $\omega = xyz$ with $|xy| \leq n$ and $|y| > 0$

Step3: Now we want to find i, such that $xy^i z \notin L, \exists i \geq 0$

The String y Consists of only a's i.e $y = a^k$ $(1 \leq k \leq n)$

$$\omega = xyz = a^n b^n = \underbrace{a^{n-k}}_{x} \underbrace{a^k}_{y} \underbrace{b^n}_{z}, \quad |xy| \leq n$$

$$xy^2z = a^{n-k} a^{2k} b^n = a^{n+k} b^n \notin L$$

Contradicts our assumption Hence L is not Regular language.

#Imp:

Ex:- Prove that $L = \{ww \mid w \in (0,1)^*\}$ is not regular language

Sol^n:-

Step1:- Assume L is regular. Let n be the no. of states in F.A.

Step2:- Let $\omega = 0^n 1 0^n 1$, $|w| \geq n$
$2n+2 > n$.

Step3: We need to show that for any Partition $\omega = xyz$ with $|xy| \leq n$ and $|y| > 0$ There exist $i \geq 0$ Such that $xy^i z \notin L$

$$\omega = xyz = 0^n 1 0^n 1 = \underbrace{0^{n-k}}_{x} \underbrace{0^k}_{y} \underbrace{1 0^n 1}_{z}, \quad |xy| \leq n$$

$$xy^2z = 0^{n-k} 0^{2k} 1 0^n 1$$
$$= 0^{n+k} 1 0^n 1 \notin L, \text{ Contradiction}$$

Contradicts our assumption Hence L is not Regular language.