Regular Grammer (Generator).

↓

Language

↓

Finite Automata (Acceptor).

Regular Grammer → Regular Language → Finite Automata

Context Free Grammer → Context Free Language → PDA (PushDown Automata

Context Senstive Grammer → Context Senstive Language → LBA Linear Bounded Automata.

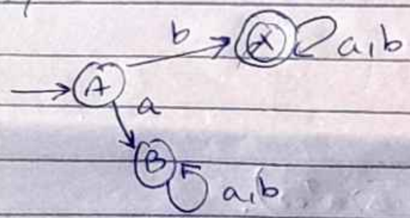
REG Recursively Enumerable Grammer → Recursively Enumerable Language → Turing Machine.

→ A Grammer is defined by a 4-Tuple

$G = \{ \Sigma, V_n, P, S \}$

Σ = finite set of Terminals / Lower case
V_n = finite non-empty set of non-terminals / upper case.
P = finite non-empty set of production rules
S = Start Symbol.

For Exp:



That begins with 'b'.

$S \rightarrow bZ$
$Z \rightarrow aZ \mid bZ \mid \epsilon$ | means 'OR'.
OR.
~~$X \rightarrow aX$~~
~~$X \rightarrow bX$~~
~~$X \rightarrow \epsilon$~~

Can be written as.
$Z \rightarrow aZ$
$Z \rightarrow bZ$
$Z \rightarrow \epsilon$.

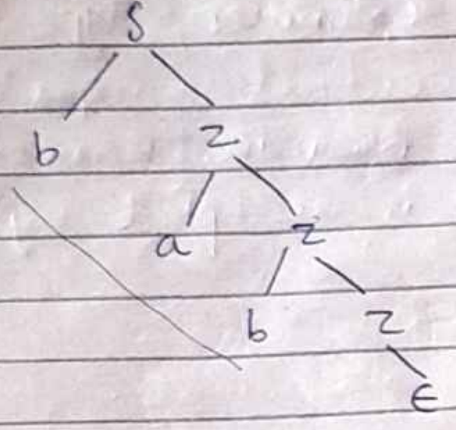
$Z \rightarrow aZ$
Terminal → cannot be changed.
non Terminal → can be changed

Here:

$\Sigma = \{a, b\}$
$V_n = \{Z\}$
$P = \{ S \rightarrow bZ, Z \rightarrow aZ \mid bZ \mid \epsilon \}$ production Rule
S = Start Symbol. (S).

Derivation (Generates strings of Language)

→ Tree Form



String: bab

→ Sentence form

$S \to bZ$
$\to baZ$
$\to babZ$
$\to bab\epsilon$
$\to \underline{bab}$.

$S \to bZ$
$\to bbZ$
$\to bbaZ$
$\to bba\epsilon$
$\to \underline{bba}$

→ Classification of Grammer

Based on Production Rules).

$\alpha \to \beta$ (production Rule).

one nonterminal necessary on α side.

$\alpha \in (\Sigma + V_n)^* V_n (\Sigma + V_n)^*$

$\beta \in (\Sigma + V_n)^*$

Valid Rule:

$aAba \to \epsilon$ ✓ where $\Sigma = \{a, b\}$

$(\Sigma + V_n)^1 A (\Sigma + V_n)^2 \to (\Sigma + V_n)^0$ $V_n = \{A\}$

Invalid:

$a \to \epsilon$ ✗.

α β

→ Recursive Enumerable Grammer / Type-0 Grammer

- Used to generate Recursive Enumerable Language (REC) which is accepted by a Turing Machine.

$$\alpha \rightarrow \beta$$

$$\alpha \in (\Sigma + V_n)^* V_n (\Sigma + V_n)^*$$
$$\beta \in (\Sigma + V_n)^*$$

→ Context Senstive / Type-1 Grammer.

- generates context sensitive Language which is accepted by LBA (Linear Bounded Automata)

$$\alpha \rightarrow \beta$$
$$\alpha \in (\Sigma + V_n)^* V_n (\Sigma + V_n)^*$$
$$\beta \in (\Sigma + V_n)^+$$
$$|\alpha| \leq |\beta|$$

Examples:

Invalid:

$A \rightarrow \boxed{\epsilon}$ ✗. should be atleast one occurrence.

$AbB \rightarrow aa$ ✗ Length of α should be less than equal to β.

→ Context Free | Type-2 Grammer.

- Used to generate Context Free Language (CFL) which is accepted by a PDA (PushDown Automata |

$\alpha \rightarrow \beta$

$\alpha \in V_n \quad |\alpha| = 1$.

$\beta \in (\Sigma + V_n)^*$.

→ Regular Gramme/ Type-3 Grammer

- generates a regular Language which is accepted by a Finite Automata.

Left Linear Grammer	Right Linear Grammer.
Non-Terminal position should be extreme left. $\underline{A \rightarrow a \mid Ba}$ Rule. $\alpha \rightarrow \beta$; $\alpha, \beta \in V_n$; $|\alpha| = |\beta| = 1$.	$A \rightarrow a \mid aB$. Non-Terminal position should be extreme right. Should not be in middle.

→ Length of non-terminal V_n should be equal to exactly 1

$A, B \in V_n \quad |A| = |B| = 1$.

Invalid

$A \rightarrow aBa$ × non-terminal is in middle.

04/04/2024.
Friday

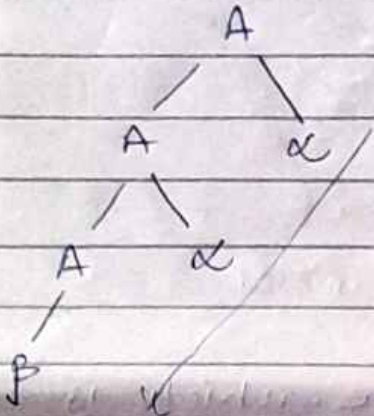
Lecture No. 24

→ Regular Grammer to Regular Expression.

i) $A \to A\alpha \mid \beta$ (Production Rule).

$A \to$ Non-Terminal

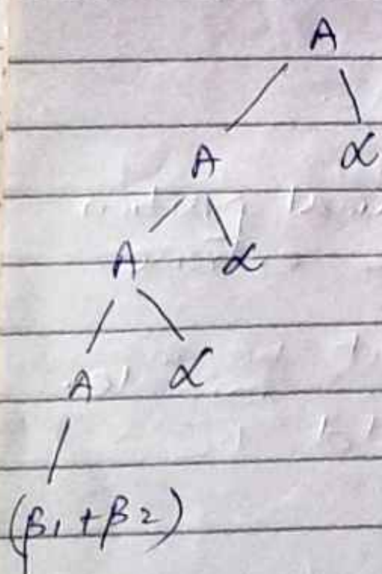
$\alpha, \beta \to$ Terminal.

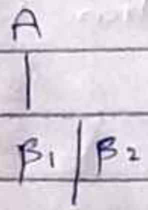


$\longrightarrow$ Regular Expression

$\beta\alpha^*$

ii) $A \to A\alpha \mid \beta_1 \mid \beta_2$



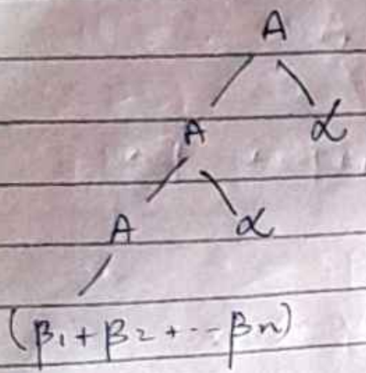


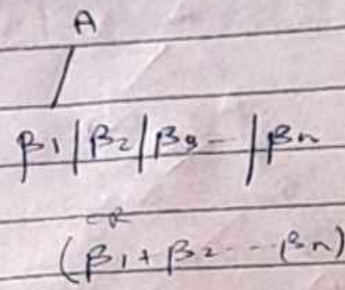
$(\beta_1 + \beta_2)$.

Regular Expression.

$(\beta_1 + \beta_2)\alpha^*$.

iii) $A \rightarrow A\alpha \mid \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$.

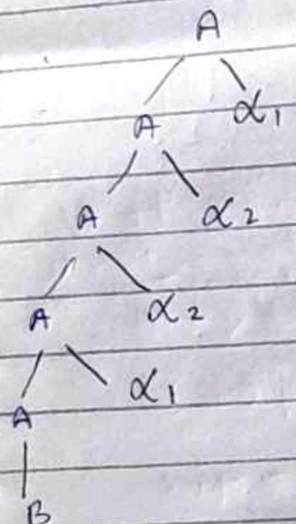




Regular Expression

$(\beta_1 + \beta_2 + \beta_3 \cdots + \beta_n)\alpha^*$.

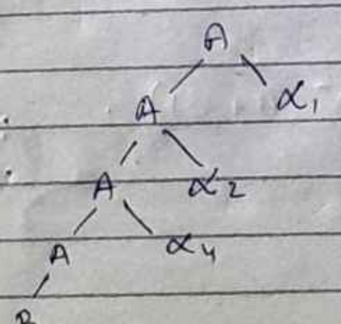
iv) $A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \beta$



Regular Expression:

$\beta(\alpha_1 + \alpha_2)^*$.

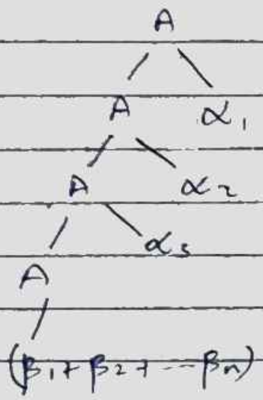
v) $A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \cdots \mid A\alpha_n \mid \beta$.

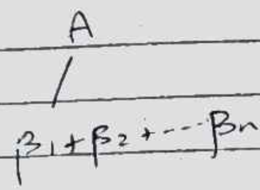


Regular Expression.

$\beta(\alpha_1 + \alpha_2 + \cdots \alpha_n)^*$

vi) $A \to A\alpha_1 \mid A\alpha_2 \mid \dots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \dots \mid \beta_m$

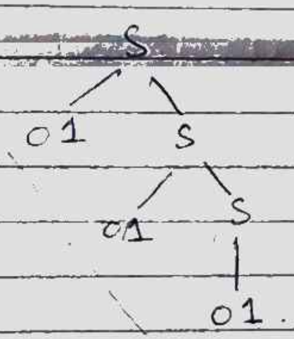




Regular Expression

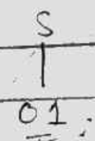
$(\beta_1+\beta_2+\dots\beta_n)(\alpha_1+\alpha_2+\dots\alpha_n)^*$

vii) $S \to 01S/01$. (Right Linear Grammar).

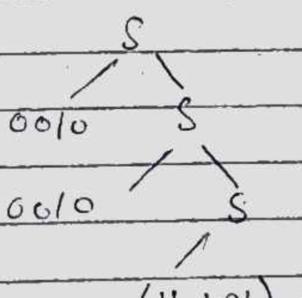


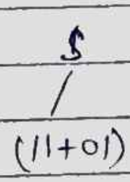
R.E:

$= (01)^* 01$.



viii) $S \to 0010S/11/01$



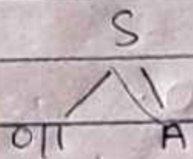


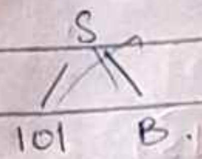
R.E:

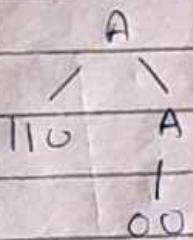
$(0010)^*(11+01)$.

(a) $S \rightarrow 011A \mid 101B$

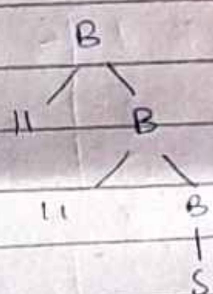
$A \rightarrow 110A \,/\, 00$, $B \rightarrow 11B \mid S$.







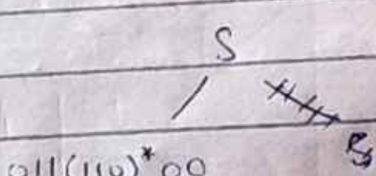
A: $(110)^* 00$

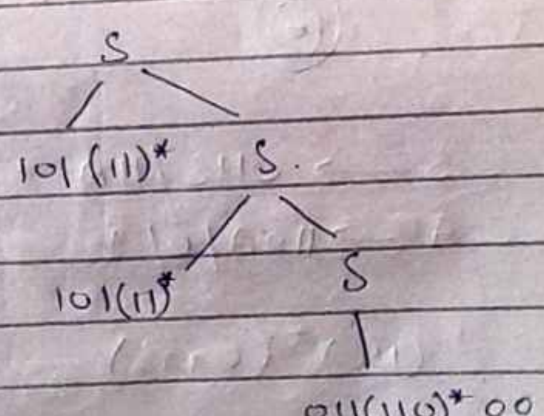


B: $(11)^* S$.

So S:

$S \Rightarrow 011(110)^* 00 \mid 101(11)^* S$.





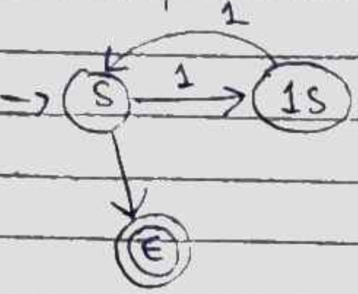
Final Regular Expression:

$\boxed{((101)(11)^*)^* (011(110)^* 00)}$. Ans.

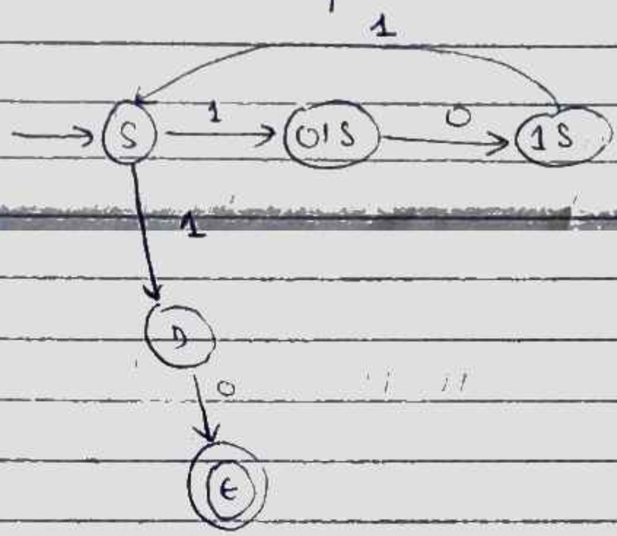
→ Regular Grammer to finite Automata.

i) $S \rightarrow 11S / 0$.

R.E: $(11)^* 0$.



ii) $S \rightarrow 101S \,/\, 10$

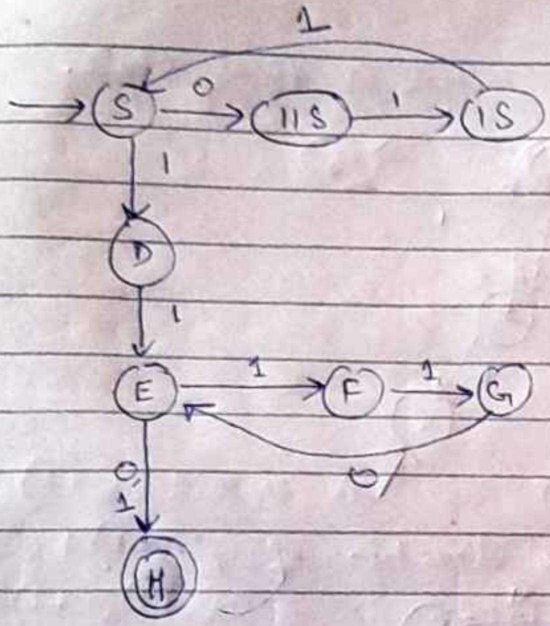


iii) $S \rightarrow 011S / 11A$

$A \rightarrow 110A / 0 / 1$

$\hookrightarrow (110)^* (0+1)$

$S \rightarrow 011S / 11(110)^*(0+1)$



07/08/24

Lecture No. 25

Friday.

→ Left Linear Grammar to finite Automata

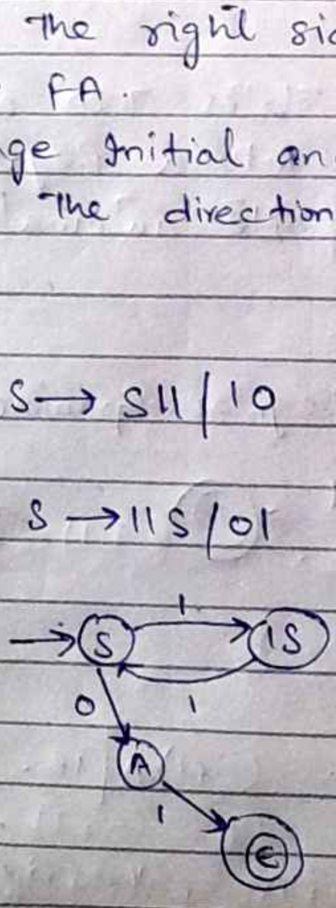
1. Reverse the right side of every production rule.
2. Construct FA.
3. Interchange initial and final state
4. Change the direction of edges.

Example :

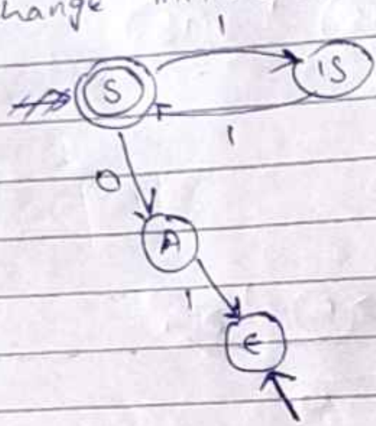
$S \rightarrow S11 \mid 10$

step 1: $S \rightarrow 11S \mid 01$

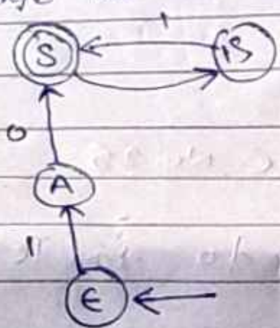
step 2:



step 3: interchange initial & final state.



step 4: change the direction of Edges.

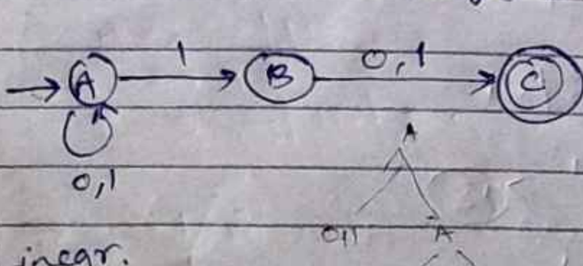


R.E : $10(11)^*$.

Note:

If multiple accepting states then convert left linear Grammer into a Regular Expression and then make finite automata.

→ Finite Automata to Regular Grammer:



2ND Last symbol is '1'.

Right Linear.

$A \rightarrow 0A|1A|1B. \Rightarrow (0+1)A|1B.$

$B \rightarrow 0C|1C \Rightarrow 0|1 - (0+1).$

$C \rightarrow \epsilon.$ Epsilon shows accepting state.

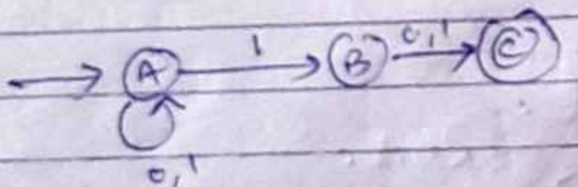
$$A \rightarrow (0+1)A$$
$$\rightarrow (0+1)(0+1)A$$
$$\rightarrow (0+1)^*A$$

So

$$(0+1)^* 1B.$$
$$\boxed{(0+1)^* 1 (0+1)} \text{ Ans.}$$

→ Left Linear Grammar:

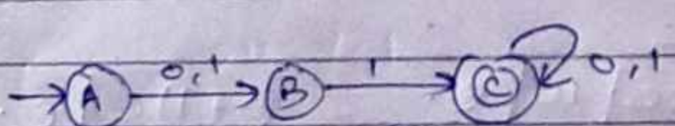
1. Obtain RE.
2. Reverse R.E.
3. Construct FA.
4. Construct Right Linear Grammar
5. Reverse the right side of every production Rule.



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

(2) $(0+1) 1 (0+1)^*$

(3)



(4) $A \rightarrow 0B \mid 1B$

$B \rightarrow 1C$

$C \rightarrow 0C \mid 1C \mid \epsilon$.

(5)

$A \to B0 \mid B1$

$B \to C1$

$C \to C0 \mid C1 \mid \epsilon$

LLG to R.E:

$A \to B(0+1) \longrightarrow A \to C1(0+1)$

$B \to C1 \qquad C(0+1)1(0+1)$

$C \to C(0+1) \mid \epsilon. \qquad \underline{C(0+1)(0+1)1(0+1)}.$

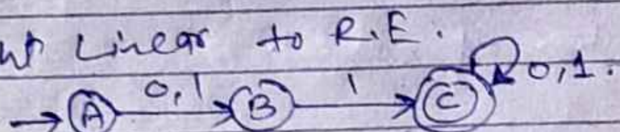
$C(0+1)^*1(0+1)$

C can be replaced with Terminal ϵ to get R.E.

$\epsilon.(0+1)^*1(0+1)$

$\boxed{(0+1)^*1(0+1)}$.

Right Linear to R.E.



~~$A \to 0A \mid 1A$~~

$A \to 0B \mid 1B$

$B \to 1C$

$C \to 0C \mid 1C \mid \epsilon. \quad - (0+1)C \mid \epsilon.$

$A \to (0+1)B$

$A \to (0+1)1C.$

$(0+1)1(0+1)C$

$(0+1)1(0+1)(0+1)C \Rightarrow \boxed{(0+1)1(0+1)^*}$