(Unit: 3)

Grammer: $G = (V_n, \Sigma, P, S)$

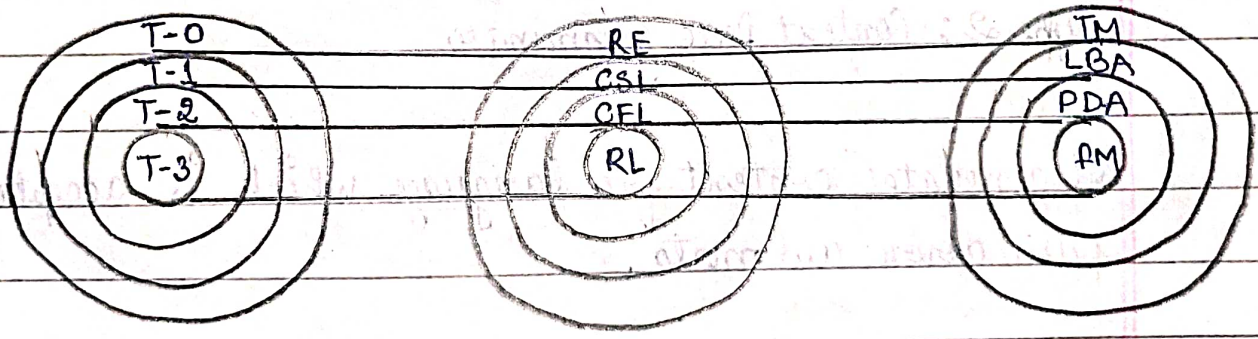
V_n = Set of non-terminals.

Σ = Set of alphabets (Terminals).

P = Production rule.

S = Starting Symbol.

Grammer: The set of all string that can be drived from the grammer is said to be the language generated from the grammer.



(GRAMMER)

FM: Finite Machine

LAB: Linear Bounded Automata

RE: Recurring Enumerable

TM: Turing Machine

PAGE NO.:
DATE: / /20

Type 0 : Recursive enumerable / unrestricted Grammer / Phased Structure Grammer.

$\alpha \to \beta$; $\alpha \in (\Sigma \cup V_n)^*$, V_n $(\Sigma \cup V_n)^*$, $\beta \in (\Sigma \cup V_n)^*$

Type 1 'Grammar' : Length increasing Grammer (Non-Contracting language). It generates context sensitive langua which is accepted by linear automata.

$$\alpha \to \beta$$
$$\alpha \in (\Sigma \cup V_n)\ V_n\ (\Sigma \cup V_n)^*$$
$$\beta \in (\Sigma \cup V_n)^*$$
$$|\alpha| \leq |\beta|$$

V_n = Set of non-terminal.

Type 2 : Context free Grammar.

It generates context free language which is accepted by push down automata.

$$\alpha \to \beta$$
$$\alpha \in (\Sigma \cup V_n)\ V_n\ (\Sigma \cup V_n)^*$$
$$\beta \in (\Sigma \cup V_n)^*$$
$$|\alpha| \leq |\beta|$$

$$\alpha \to \beta$$
$$\alpha \in V_n \quad |\alpha| = 1.$$
$$\beta \in (\Sigma \cup V_n)^*$$

Type 3 Grammer : Regular Grammer.

It generates regular language which is accepted by finite automata.

Left linear grammer	Right linear Grammer
$A \rightarrow a/Ba$.	$A \rightarrow a/aB$.
$A, B \in V_n$	$A, B \in V_n$.
$\|A\| = \|B\| = 1$	$\|A\| = \|B\| = 1$.

Ques 1) $G = \{\{S\}, \{0,1\}, \{S \rightarrow 0S1\}, \{S \rightarrow \Lambda\}\}$ Find $L(G)$.

Soln

$S \rightarrow 0S1$

$\rightarrow 00S11$

$\rightarrow 000S111$

$\rightarrow$ -----

$\rightarrow 0^n 1^n$

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

Ques 2) $G = \{\{S\}, \{a\}, \{S \rightarrow SS\}, S\}$. Find language generated by the Grammer

$S \rightarrow SS$.

$L(G) = \phi$

Ques 3) $G = \{(S,C), (a,b), P, S\}$ $S \rightarrow aCa$.

$C \rightarrow aCa/b$

Find $L(G)$.

Soln

$S \to aca$.

$S \to aba \in L$.

$S \to aca$.

$S \to aacaa$

$S \to aaacaaa$

$\to a^n c a^n$

$\to a^n b a^n$

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

Find $L(G)$.

Sol^n

$S \to aCa$

$S \to aba \in L$

$S \to aCa$

$S \to aaCaa$

$S \to aaaCaaa$

$\to a^n C a^n$

$\to a^n b a^n$

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

Derivation Tree: A derivation tree or Parse tree is an ordered, rooted tree, that rapidly represent the semantic information of strings derived from a Context Free Grammar.

Example: For the Grammar

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

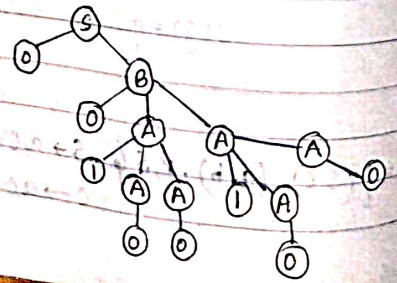
$S \to 0B$, $A \to 1AA$, $B \to 0AA$

T = Set of Terminals

P = Production Rule

S = Start

V = Set of Variables



Root Vertex: Root Vertex must be levelled by start Symbol

Vertex: Vertex is levelled by non-terminal Symbol

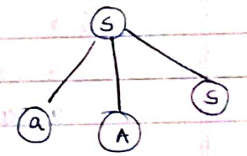
Leaves: Leaves are levelled by terminal Symbol or ε

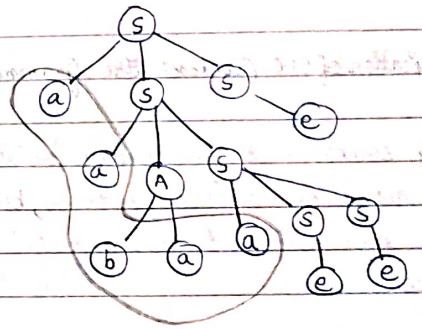
Left Derivation Tree: A left Derivation tree is obtained by a prime production of the left most variable in each step.

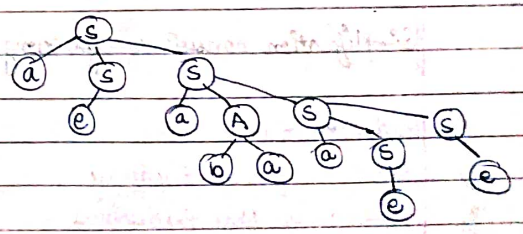
For example: For generating the starting symbols (aabaa) from the grammar

$S \to aAS \mid aSS \mid \varepsilon$.

$A \to sbA \mid ba$







Ambiguous: A grammar is said to be ambiguous if there exist two or more derivation tree for a string w that means two or more left derivation tree.

Ques) $G = \{S\}, \{a+b, +, \times\}, P, S$

P consist of ;

$$S \rightarrow S+S \mid S*S \mid a \mid b$$

Sol^n

$$W = a+a*b$$

$S \rightarrow S+S$ $\quad$ $S \rightarrow S*S$

$S+S*S.$ $\quad$ $S \rightarrow S+S*S$

$a+a*b$ $\quad$ $a+a*b$

Simplification of CFG (Context Free Grammar):

In CFG, sometimes all the production tools and symbols are not needed for the derivation of strings, besides this, there may also be some null productions and Unit productions and symbols.

Simplification consists of following steps:

1) Reduction of CFG
2) Removal of Unit Productions
3) Removal of Null Productions

1) Reduction of CFG (Phase 1): Derivation of an Equivalent Grammar G' from the CFG (G) such that each symbol appears in sentential form.

Derivation Procedure: * Include the start symbol Y_1 and initialize $i = 1$

* Include all symbols Y_{i+1} that can be derived from Y_i and include all production rules that have been applied.

* Increament i, and repeat Step 2, until $Y_{i+1} = Y_i$

Example: Find the reduced grammar equivalent to the grammar G having production rules.

P: $S \to AC \mid B$, $A \to a$, $C \to c \mid Bc$, $E \to aA \mid e$

Phase 1: $T = \{a, c, e\}$

$\omega 1 = \{A, C, E\}$

$\omega 2 = \{A, C, E, S\}$

$\omega 3 = \{A, C, E, S\}$

$G' = \{(A, C, E, S), (a, c, e), P, S\}$

P: $A \to a$, $C \to c$, $E \to e$, $S \to AC$, $E \to aA$

PAGE NO.:
DATE 26/04/2024

Phase 2: $Y_1 = \{S\}$

$Y_2 = \{S, A, C\}$

$Y_3 = \{S, A, C, a, e\}$.

$Y_4 = \{S, A, C, a, e\}$.

$q'' = \{(A,C,S), (a,c), P, \{S\}\}$

$S \to AC$, $A \to a$, $C \to c$

Chomsky Normal Form: In Chomsky Normal Form, we have a restriction on the length of RHS (Right Hand Side) which is elements in RHS, should either be 2 variables or a Terminal.

$A \to a$

$A \to BC$

Steps to convert in CNF: * If the start symbol 'S' occurs on Right Side, create a new start Symbol 'S', and a new product ($S' \to S$)

* Remove Null productions.

* Remove Unit production ($S' \to S$) [Eg. $A \to B$]

* Replace each production (Eg: $A \to B_1, B_2, \ldots\ldots B_n$) where, $n > 2$ with $A \to B_1 C$, where $C \to B_2, \ldots B_n$

PAGE NO.:
DATE: / /20

★ Repeat this step for all productions, having 2 or more symbols, on the right side. If the right side of all production is in the form of $A \rightarrow aB$, then production is replaced by $A \rightarrow XB$ and $X \rightarrow a$.

★ Repeat this step, for every production which is of the form $A \rightarrow aB$.