A grammar can be represented by four tu[ples]
V T P S.
V – set of all variables / auxilliary symbol / non-terminal symbol
T – set of all terminal
P – set of all production
S – start symbol

$$P = \left\{ \begin{array}{l} S \rightarrow aSB \\ S \rightarrow aB \\ B \rightarrow b \end{array} \right\} \quad \begin{array}{l} V \ \{S, B\}. \\ T \ \{a, b\}. \end{array}$$

Ques.

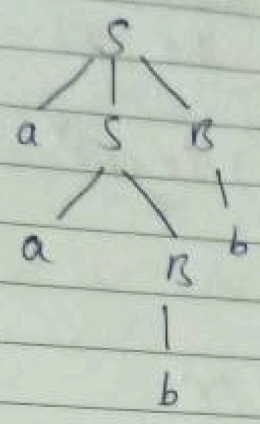
Derivation :- Deriving a string from the gram[mar] starting from the start symbol.

aabb

$$\begin{aligned} S &\rightarrow aSB \\ &\rightarrow aaBB \\ &\rightarrow aabB \\ &\rightarrow aabb \end{aligned}$$

- At every step only one variable will be repl[aced] by RHS.
- If we start replacing the left most symbol first then it is called left most derivation.
- If we start replacing the right most symbol first then it is called right most derivati[on].
- Entire process is called derivation of strin[g].
- Intermediate steps are known as sentencial form or sequential form.
- One of the way of representing the derivatio[n]

is called derivation tree or parse tree.



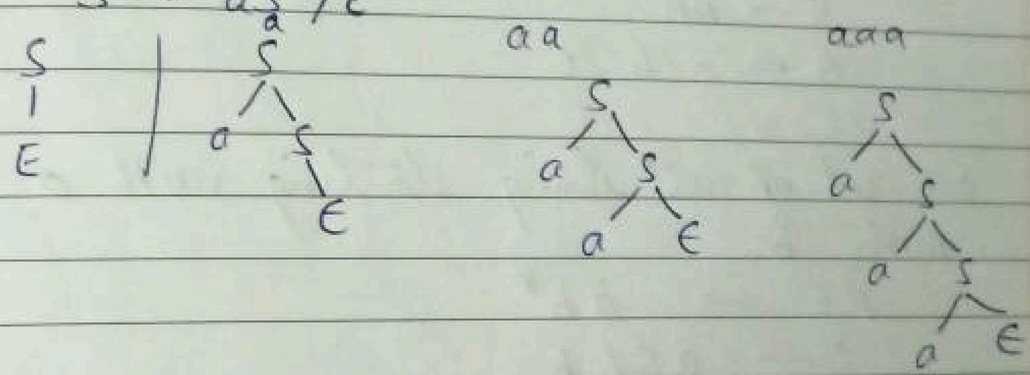
aabb

Ques. $L = a^n$ such that $n \geq 0$.

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

$S \rightarrow aS \,/\, \epsilon$

aa　　　　aaa



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

$S \rightarrow aS \,/\, a$

$L = (a+b)^*$

$S \rightarrow aS / bS / \epsilon$

Q. L = set of all string of length at least two
L = {aa, aaa, aaaa, ... }
{ aa, ab, ba, bb

$RE = \underbrace{(a+b)}_{A}(a+b)\underbrace{(a+b)^*}_{B}$

$S \rightarrow AAB$
$A \rightarrow a/b$
$B \rightarrow aB/bB/\epsilon$

Q. L = set of all string of length at most two
$(a+b+\epsilon)(a+b+\epsilon)$
$S \rightarrow AA$
$A \rightarrow a/b/\epsilon$

Q. L = set of all string starting with a ending with b.
$RE \rightarrow ab^*$
$S \rightarrow aS/b$

$RE \rightarrow a(a+b)^*b$
$S \rightarrow aAb$
$A \rightarrow aA/bA/\epsilon$

L = set of all string starting with b ending w

$RE \rightarrow b(b+a)^*a$
$S \rightarrow bAa$
$A \rightarrow bA/aA/\epsilon$

Ques. L = set of all string starting & ending with different symbol.

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

$S \rightarrow aAb \mid bAa$

$A \rightarrow aA \mid bA \mid \epsilon$

Ques. L = set of all string starting and ending with same symbol.

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

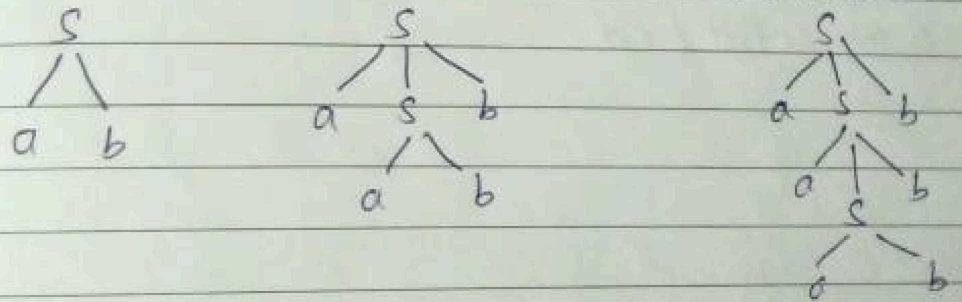
$S \rightarrow aAa \mid bAb \mid a \mid b \mid \epsilon$

$A \rightarrow aA \mid bA \mid \epsilon$

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

$S \rightarrow aSb \mid ab$

ab　　aabb　　aaabbb



$ww^R \cup wawR \cup wbw^R,\ w \in \{ab\}^*$

$S \rightarrow aSa \mid bSb$ – even

$S \rightarrow aSa \mid bSb \mid a \mid b$ – odd

$S \rightarrow aSa \mid bSb \mid a \mid b \mid \epsilon$ – all

Ques. Construct a grammar over $\Sigma = \{a, b\}$ of even length string.

$RE = \underbrace{(\underbrace{(a+b)}_{A}\underbrace{(a+b)}_{A})}_{B}{}^*$

$S \rightarrow BS \mid \epsilon$

$B \rightarrow AA$, $A \rightarrow a \mid b$

Ques. $L = \underbrace{a^n}_{A}\,\underbrace{b^m}_{B} \mid n, m \geq 1$

$S \rightarrow AB$

$B \rightarrow bB / b$

$A \rightarrow aA / a$

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

$S \rightarrow AB$

$B \rightarrow cB / c$

$A \rightarrow aAb / ab$

Ques. $L = \underbrace{a^n b^n}_{A}\,\underbrace{c^m d^m}_{B} \mid n, m \geq 1$

$S \rightarrow AB$

$A \rightarrow aAb / ab$

$B \rightarrow cBd / cd$

Classification of Grammar :- A/C to chomsky grammar are classified into four type

Type 3 grammar → regular grammar

Type 2 → Context free grammar

Type 1 → Context

Type 0 → Unrestricted grammar

Type 3 :-

Right linear grammar	Left linear grammar
$A \rightarrow \alpha B / \beta$	$A \rightarrow B\alpha / \beta$
$\{A, B\} \in V$	$\{A, B\} \in V$
$\{\alpha, \beta\} \in T^*$	$\{\alpha, \beta\} \in T^*$
Ex:- $A \rightarrow aB / a$	Ex:- $A \rightarrow Ba / a$
$B \rightarrow aB / bB / a / b$	$B \rightarrow Ba / Bb / a / b$

Ex:- $A \rightarrow Ba / a$
$B \rightarrow aB / a$

This isn't Type 3 grammar because one is LLG & one is RLG

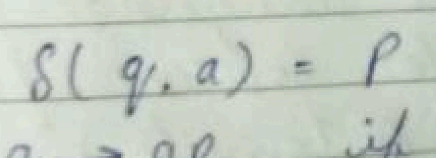
$\delta(q, a) = p$

$q \rightarrow ap$ if q isn't a final state

$\delta(q, a) = p$

$q \rightarrow ap / \epsilon$ if q is a final state

FA → RLG

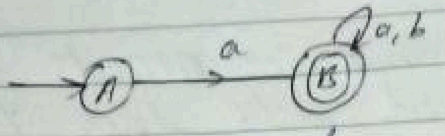


$A \rightarrow aB$

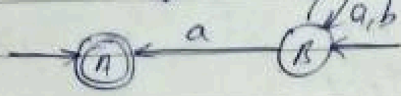
$B \rightarrow aB / bB / \epsilon$

$FA \longrightarrow LLG$

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

Ques. 

$\downarrow$ Reversal



$\left.\begin{array}{l} B \rightarrow aB/bB/aA \\ A \rightarrow \epsilon \end{array}\right\} RLG$

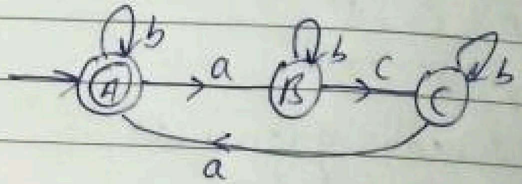
$\left.\begin{array}{l} B \rightarrow Ba/Bb/Aa \\ A \rightarrow \epsilon \end{array}\right\} LLG$

$RLG \longrightarrow FA$

$A \rightarrow aB/bA/b$

$B \rightarrow cC/bB$

$C \rightarrow aA/bC/a$



Type 2 :— If production are of the form $A \rightarrow \alpha$ where $A \in V$, $\alpha \in (V \cup T)^*$, then such a grammar is called type 2 grammar or context free grammar.

Languages generated by context free grammar also known as context free language and machine use as acceptor for context free language is PDA (push down Automata).

$E \rightarrow E+E / E*E / id$

$V = \{E\}$

$T = \{+, *, id\}$

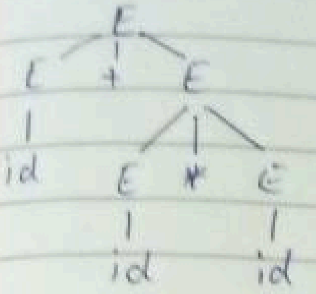
$E \rightarrow E+E$ $\qquad\qquad$ $id + id * id$

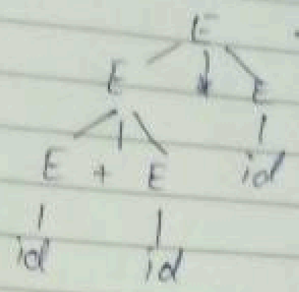
$\rightarrow E + E * E$

$\rightarrow id + E * E$

$\rightarrow id + id * E$
$\rightarrow id + id * id$

$E \rightarrow E * E$
$\rightarrow E + E * E$
$\rightarrow id + E * E$
$\rightarrow id + id * E$
$\rightarrow id + id * id$

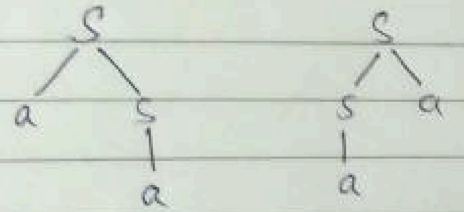




for a given string if more than one of LMD (Left most derivation), RMD, parse tree exist then we can say that grammar is ambiguous

Q. $S \rightarrow aS \,/\, Sa \,/\, a$

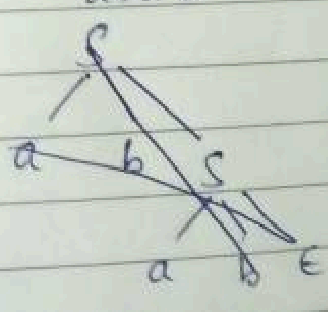
aa

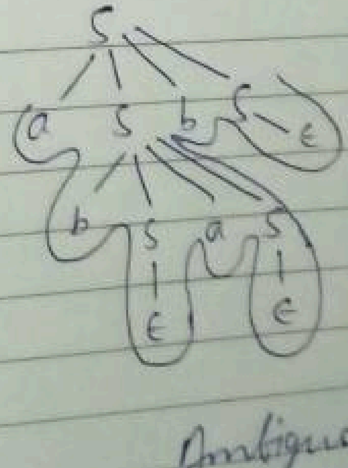


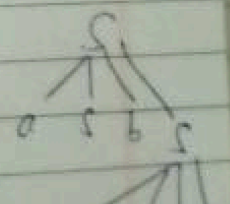
Ambiguous

$S \rightarrow aSbS \,/\, bSaS \,/\, \epsilon$

abab







Ambiguous