Assignment

1 a] A grammar G is 4-tuple or quadruple $G=(V, T, P, S)$ where

i] V is set of variables

ii] T is set of terminals

iii] P is set of production each production is of the form $\alpha - \beta$ where $\alpha \in (V \cup T)^+$ [does not contain ϵ] and $\beta \in (V \cup T)^*$ [contains ϵ]

iv) S is the start symbol.

Example: $S \rightarrow aB \mid bA \mid \epsilon$

$A \rightarrow aA \mid b$

$B \rightarrow bB \mid a \mid \epsilon$

b] Derivation: The process of obtaining strings of terminals and/or non-terminals from the start symbol by applying some or all productions is called derivation & The derivation to get string id + id * id is

Derivation 1→ $E \rightarrow E+E$

$\rightarrow id + E$

$\rightarrow id + E * E$

$\rightarrow id + id * E$

$\rightarrow id + id * id$

c] Derivation Tree: Let $G=(V, T, P, S)$ be a CFG. The tree is derivation tree with following properties:

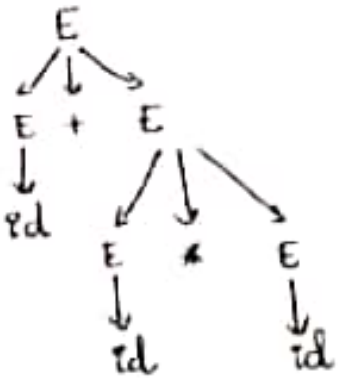
1] The root has label S

2] Every vertex has a label which is in $(V \cup T \cup \epsilon)$

3] Every leaf node has label from T and an interior vertex has a label from V.

4] If a vertex is labelled A and if $X_1, X_2, X_3 \ldots X_n$ are all children of A from left, then $A \rightarrow X_1 X_2 X_3 \ldots X_n$ must be a production in P

Derivation tree for (1)



3] The grammar to generate strings of palindromes over Σ is given by $G = (V, T, P, S)$ where

$V = \{S\}$

$T = \{a, b\}$

$P = \{$

$S \to \varepsilon$ [ε is a palindrome]

$S \to a \mid b$ [a and b are palindromes]

$S \to aSa \mid bSb$ [If w is a palindrome then the strings awa and bwb are palindromes]

$\}$

S is the start symbol.

5] $V = \{S\}$

$T = \{0, 1\}$

$P = \{$

$(S \to 0S1 \mid \varepsilon$ [The grammar used to generate equal number of 0's and 1's])

For the grammar which should generate one extra 0 compared to 1 we have

$S \to 0 \mid 0S1$

$\}$

S is start symbol.

4. The complete grammar to generate an arithmetic expression is

$V = \{E, I\}$

$T = \{+, -, *, /, ^, a, b, c\}$

$P =$

$E \rightarrow I$ → [An expression E can be an Identifier]

a. $E \rightarrow E + E$
b. $E \rightarrow E - E$
c. $E \rightarrow E * E$
d. $E \rightarrow E / E$
e. $E \rightarrow E \wedge E$
f. $E \rightarrow (E)$

→ [If E is any arithmetic expression then a to f are productions of arithmetic expressions]

$I \rightarrow Ia \mid Ib \mid Ic \mid a \mid b \mid c$ → [An identifier of length at least one can be generated using any combination of a's, b's and c's]

S

S is the start symbol

5] The grammar for equal number of a's and b's are

$A \rightarrow aAb$

$A \rightarrow bAa$

$A \rightarrow AA$

$A \rightarrow \varepsilon$

Since number of a's should be more when compared to b's we should generate one or more a's. The grammar to obtain it is

$B \rightarrow aB \mid a$

Any no of a's can occur in the beginning, middle or end. Equal no of a's and b's can be followed by one or more a's at respective places by introducing production

$S \rightarrow AB$

Equal no of a's and b's can be preceded by one or more a's by production

$S \rightarrow BA$

equal number of a's and b's can have one or more a's in middle step introducing production

$S \rightarrow ABA$

The grammar is

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

$T = \{a, b\}$

$P = \{$

$S \rightarrow AB \mid BA \mid ABA$

$A \rightarrow aAb$

$A \rightarrow bAa$

$A \rightarrow AA$

$A \rightarrow E$

$B \rightarrow aB \mid a$

$\}$

S is start symbol.

i] The process of obtaining string of terminals and / or non-terminals from the start symbol by applying some or all productions is called derivation.

8] Leftmost derivation. In the derivation process if a left most variable is replaced at every step then the derivation is said to be leftmost derivation

Ex: $E \to E+E$
$\to id+E$
$\to id+E*E$
$\to id+id*E$
$\to id+id*id$

Rightmost derivation. In the derivation process if a right most variable is replaced at every step then the derivation is said to be rightmost derivation

$E \to E+E$
$\to E+E*E$
$\to E+E*id$
$\to E+id*id$
$E \to id+id*id$

9] A grammar G [$G_1 = (V, T, P, S)$ be a context free grammar] is ambiguous if and only if there exists at least one string $\omega \in L$]* for which two or more different parse trees exist by applying either the left most derivation or right most derivation

Ex: ① $E \to E+E$
$\to id+E$
$\to id+E*E$
$\to id+id*id$

② $E \to E*E$
$\to E+E*E$
$\to id+E*E$
$\to id+id*E$
$\to id+id*id$

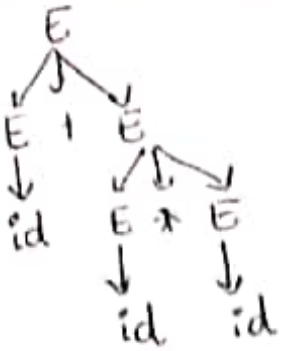
← $E \to E+E$
$E \to E-E$
$E \to E*E$
$E \to E/E$
$E \to (E) / 9$
$9 \to id$

↓ It is ambiguous

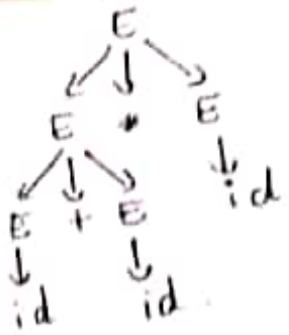
Leftmost derivation is applied

Corresponding derivation trees are

①



②



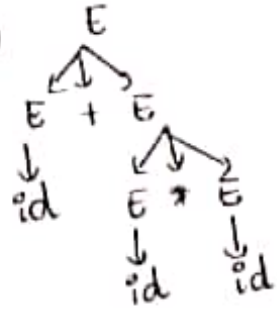
10] The sentence id + id * id can be obtained from leftmost derivation in two ways

$E \to E + E$
$\to id + E$
$\to id + E * E$
$\to id + id * E$
$\to id + id * id$

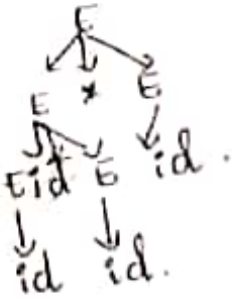
$E \to E * E$
$\to E + E * E$
$\to id + E * E$
$\to id + id * E$
$\to id + id * id$

The corresponding derivation trees are

①



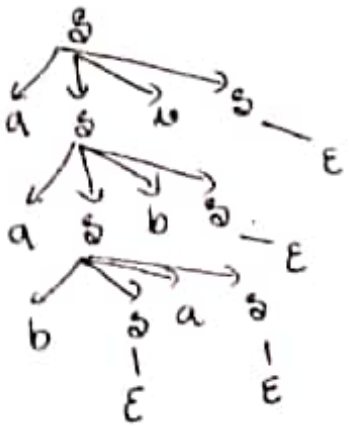
②



Since derivation trees are different grammar is ambiguous.

11] Consider the leftmost derivation for string aabbabb and the corresponding parse tree

$S \to aSbS$ $(S \to aSbS)$
$\to aaSbSbS$ $(S \to aSbS)$
$\to aabSaSbSbS$ $(S \to bSaS)$
$\to aabaSbSbS$ $(S \to \varepsilon)$.
$\to aababSbS$ $(S \to \varepsilon)$
$\to aababbS$ $(S \to \varepsilon)$
$\to aababb$. $(S \to \varepsilon)$



Consider the left most derivation of aabaabb but using different set of rules

$S \to aSbbS$ $(S \to aSbS)$

$\to aaSbbbS$ $(S \to aSbS)$

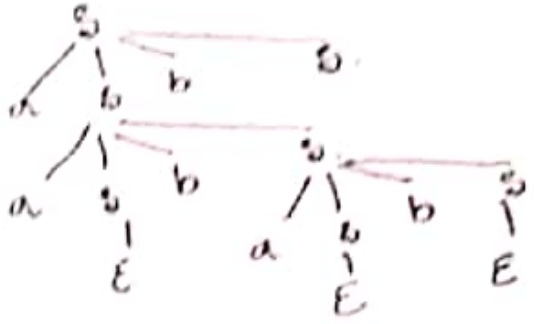
$\to aabSbbS$ $(S \to \varepsilon)$

$\to aabaSbbSbS$ $(S \to aSbS)$

$\to aabaSbbbS$ $(S \to \varepsilon)$

$\to aababbbS$ $(S \to \varepsilon)$

$\to aababbb$ $(S \to \varepsilon)$



Since there are two parse trees for the string aababb by applying leftmost derivation, the grammar is ambiguous.

10] Context - free grammars arise in linguistics where they are used to describe the structure of sentences and words in a natural language, and they were in fact invented by Chomsky for this purpose, but have not really lived up to their original expectation. By contrast, in computer science, the use of recursively - defined concepts increased, they were used more and more.

1] In an early application, grammars are used to describe the structure of programming languages.

2] In a newer application, they are used in an essential part of extensible markup language called the Document.

Applications → Parsers, YACC Parser, Markup languages, XML
→ Finite automaton
→ Digital Design.