Content - Free Grammar (CFG)

Used to generate all possible patterns of strings in a given language.

CFG → 4 tuples

$G_1 = (V, T, P, S)$

where,

G :- Grammer, set of production rules

T :- Terminal symbols (a-z)

V :- Non Terminal symbols (A-Z)

S :- Start Symbol

P :- set of production rules

→ The concept of grammer for language is starting from start symbol using production rule, deriving the string.

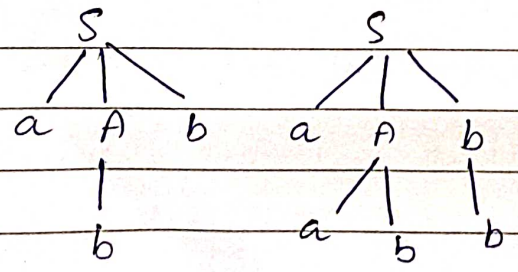
→ Productions LHS is replaced by RHS.

Example Find language :-

$S \rightarrow aAb$

$A \rightarrow aB / b$

$B \rightarrow c$



= abb

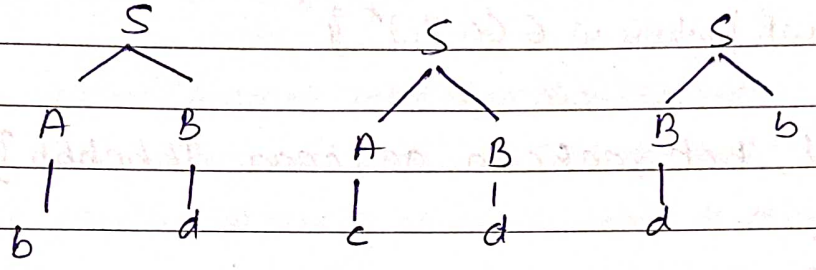
= aacb

Example Find language :-

$S \rightarrow AB / Bb$

$A \rightarrow b / c$

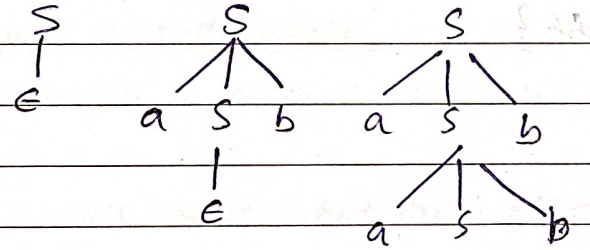
$B \rightarrow d$



$= bd$ $= cd$ $= db$

Example $S \rightarrow aSb / \epsilon$

$S \rightarrow aSb / \epsilon$



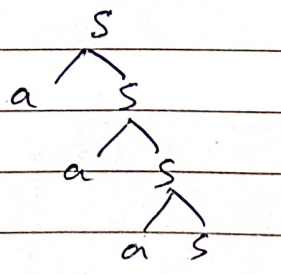
$= \epsilon$ $= ab$ $= aabb$

$L = \{ a^n b^n \mid n \geq 0 \}$

Example any no. of a's

$S \rightarrow aS$

$S \rightarrow \epsilon$



aaa

Example $(0+1)^*$

$S \rightarrow 0S \mid 1S$

$S \rightarrow \in$

Example $L = \{wcw^R \mid \text{where } w \in (a,b)^*\}$

$L = \{acat, bcb, aabcaa, aaacaaa, bbbcbbb\}$

$S \rightarrow asa$

$S \rightarrow bsb$

$S \rightarrow c$

Example $L = a^n b^{2n}$ where $n >= 1$

$L = \{abb, aabbbb, aaabbbbbb\}$

$S \rightarrow asbb \mid abb$

$\not{\leftrightarrow}$

Derivation

Derivation is a sequence of production rules.
used to get input string through these rules.

During passing we have to take two decisions →

1. Decide the non-terminal that is to be replaced
2. Decide the production rule by which the non-terminal will be replaced.

Left Most derivation :-

The input is scanned and replaced with the production rule from left to right.

we read input string from left.

Right Most derivation :-

The input is scanned and replaced with the production rule from right to left.

we read input from right to left

Example derive the string abb by →

Given rules :-

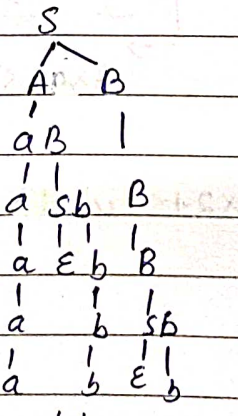
Production

$S \rightarrow AB \mid \varepsilon$

$A \rightarrow aB$

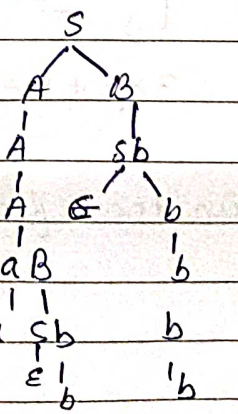
$B \rightarrow Sb$

Left Most Derivation



= abb

Right Most Derivation



abb

Ambiguity in Grammer

A grammer is said to be ambiguous if there exist ~~one~~ more than one left or right most derivative tree.

If grammer is ambiguous then it is not good for compiles construction.

Example $E \rightarrow I$

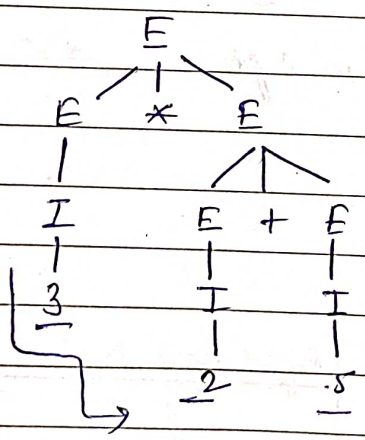
$E \rightarrow E+E$

$E \rightarrow E*E$

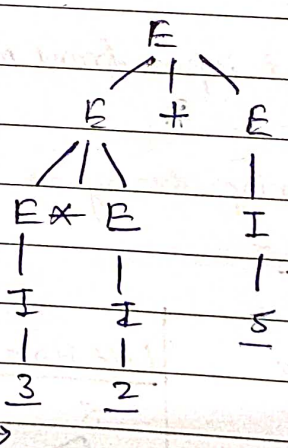
$E \rightarrow (E)$

$I \rightarrow \varepsilon \mid 0 \mid 1 \mid 2 \mid \ldots \mid 9 \mid$

$\Rightarrow$ For string $3*2+5$, the above grammer can generate two parse tree



$3 \times 2 + 5 = 11$



$3 \times 2 + 5 = 11$

Two trees for grammer $3 \times 2 + 5$ so it's ambiguous.

Example $E \to E+E$

$E \to E-E$

$E \to id$

⇒ Left most derivation

$E \to E+E$

$\to \underline{id}+E$

$\to \underline{id}+E-E$

$\to \underline{id}+\underline{id}-E$

$\to \underline{id}+\underline{id}-\underline{id}$

Right most derivation

$E \to E-E$

$\to E+E-E$

$\to \underline{id}+E-E$

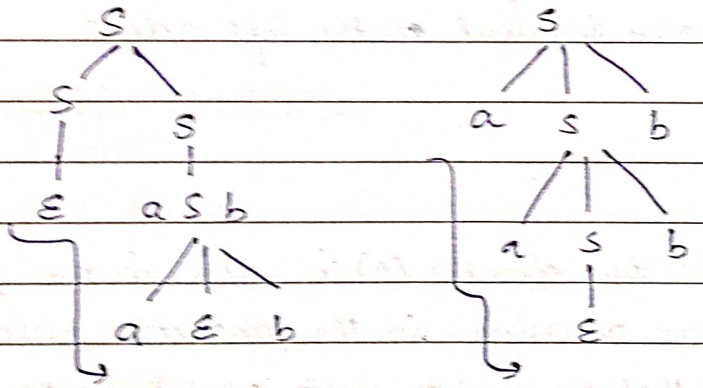
$\to \underline{id}+\underline{id}-E$

$\to \underline{id}+\underline{id}-\underline{id}$

It is ambiguous

Example $S \to aSb \mid SS$

$S \to \epsilon$



⇒ aabb ⇒ aabb

Grammar is ambiguous.

Unambiguous Grammer

A grammer is unambiguous if the grammer does not contain ambiguity that means it does not contain more than one left most or right most derivation or more than one parse tree for given input string.

To convert ambiguous grammer to unambiguous grammer apply rules :-

1. If the left associative operator (+, −, *, /) are used in the production rule, then apply left recursion in the production rule.

   Left recursion means that the left most symbol on the right side will be same as the non-terminal on the left side :-

   $X \rightarrow Xa$

2. If the right most associative operater (^) is used in the production rule then apply right recursion in the production rule.

   Right recursion means that the right most symbol on the left side is the same as the non-terminal on the right side :-

   $X \rightarrow aX$

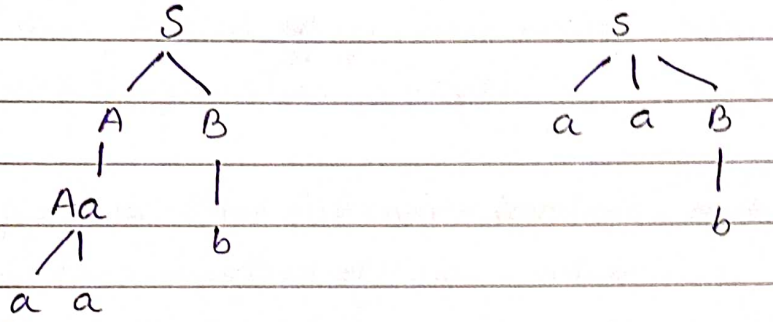
Example $S \to AB \mid aaB$

$A \to a \mid Aa$

$B \to b$

determine if grammer is ambiguous or not, if ambiguous then convert it into unambiguous.

String {aab}



$\Rightarrow aab$

Unambiguous banner :-

$S \to AB$

$A \to Aa \mid a$

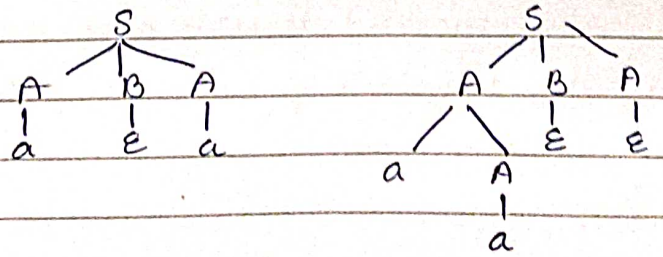
$B \to b$

Example $S \to ABA$

$A \to aA \mid \varepsilon$

$B \to bB \mid \varepsilon$

String aa



Unambiguous Grammer :-

$S \to aXY \mid bYZ \mid \varepsilon$

$Z \to aZ \mid a$

$X \to aXY \mid a \mid \varepsilon$

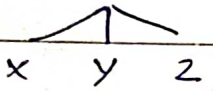
$Y \to bYZ \mid b \mid \varepsilon$

Pumping Lemma for Regular Grammer

Theorem :-

Let L be a regular language. Then there exists a constant 'c' such that for every string w in L -

$|W| \geq c$



$|y| > 0$

$|xy| \leq c$

For all $k \geq 0$, the string $xy^k z$ is also L

Application :-

Applied to show certain languages are not regular. It should never be used to show a language is regular.

→ If L is regular, it satifies Pumping Lemma.

→ If L does not satisfy Pumping lemma, it is not regular.

Method to prove language is not regular

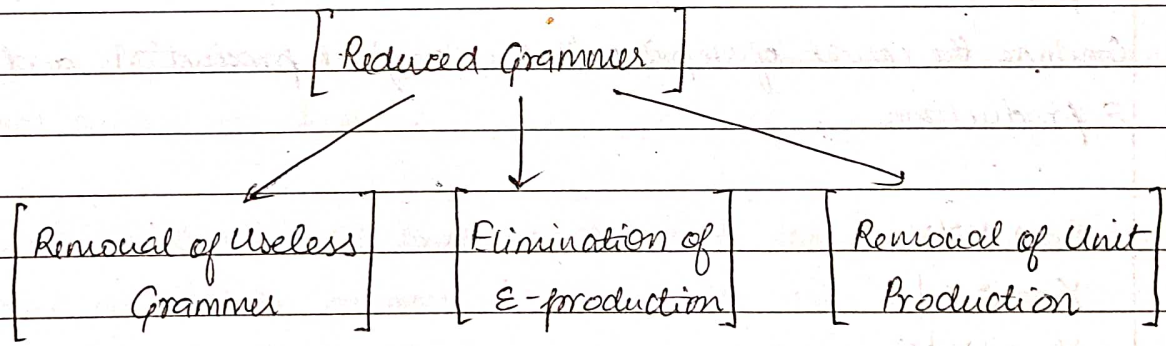
1. At first, assume that L is regular
2. So, the pumping lemma should hold for L
3. Use the pumping lemma to obtain a contradiction.
   - ↳ Select w such that $|w| \geq c$
   - ↳ Select y such that $|y| \geq 1$
   - ↳ Select x such that $|xy| \leq c$
   - ↳ Assign the remaining string to z.
   - ↳ Select k such that the resulting string is not L.

Simplification of CFG

Sometimes all the grammers are not always optimized that means the grammer may consist of some extra symbols (non-terminal). Having extra symbols, unnecessary increase the length of grammer.

Simplification of grammer means reduction of grammer by removing useless symbols.

1. Each variable (i.e non-terminal) and each terminal of G appears in the derivation of some word in L.
2. There should not be any production as $X \rightarrow Y$ where X and Y are non-terminals.
   If ε is not in the ~~production~~ language L, there is no need for production $X \rightarrow \varepsilon$



Removal of Useless Symbol

1. A symbol is useless if it doesn't appear on RHS of derivation.
2. A variable is useless if it doesn't take part in derivation

Here,

Example $T \rightarrow aaB / abA / aaT$

$A \rightarrow aA$

$B \rightarrow ab / b$

$C \rightarrow ad$

1. Here C is useless as we can accept C in any production rule
2. A is useless as its non terminating.

Elimination of ε Production

The production of type $S \rightarrow \varepsilon$ are ε-production. These productions can only be removed from those grammers that do not generate ε.

1. First find out all nullable non-terminal variable which derives ε
2. For each production $A \rightarrow a$, construct all production $A \rightarrow x$, where x is obtained from a by removing one or more non-terminal from step 1.

Combine the result of step 2 with the original production and remove ε productions.

Example- $S \rightarrow XYX$

$X \rightarrow 0X / \varepsilon$

$Y \rightarrow 1Y / \varepsilon$

$S \rightarrow XYX$

$S \rightarrow YX$ when $X_1 = \varepsilon$ (First x on RHS)

$S \rightarrow XY$ when $X_2 = \varepsilon$ (last x on RHS)

$S \rightarrow XX$ when $Y = \varepsilon$

$S \rightarrow X$ when Y and X_2 are ε

$S \rightarrow Y$ when both X are ε

$S \rightarrow XY \mid YX \mid XX \mid X \mid Y$

$X \rightarrow 0X$

$X \rightarrow 0 \quad X = \varepsilon$

$X \rightarrow 0X \mid 0$

$Y \rightarrow 1Y$

$Y \rightarrow 1$

$Y \rightarrow 1Y \mid 1$

Collectively,

$S \rightarrow XY \mid YX \mid XX \mid X \mid Y$

$X \rightarrow 0X \mid 0$

$Y \rightarrow 1Y \mid 1$

Removing Unit Productions :-

The unit productions are the productions in which one non-terminal gives another non-terminal.

1. To remove $X \rightarrow Y$, add production $X \rightarrow a$ to the grammer rule whenever $Y \rightarrow a$ occurs in the grammer.
2. Now delete $X \rightarrow Y$

Repeat 1 and 2

Example

$S \rightarrow 0A \mid 1B \mid C$

$A \rightarrow 0S \mid 00$

$B \rightarrow 1 \mid A$

$C \rightarrow 01$

$S \to C$ is a unit production. But while removing C consider C gives 01 so add rule to S

$S \to 0A|1B|C$
$S \to 0A|1B|01$

Similarly BA is a unit production.

$B \to 1|0S|00$

Final →
$S \to 0A|1B|01$
$A \to 0S|00$
$B \to 1|0S|00$
$C \to 01$

Chomsky's Normal Form (CNF)

A CFG is CNF if all productions rules satisfy one of the following conditions.

1. Start symbol generating ϵ. For $A \to \epsilon$
2. A non-terminal generating two non-terminals. For $S \to AB \mid \epsilon$
3. A non-terminal generating a terminal. For $S \to a$.

Example Convert CFG to CNF :-

$S \to ABa$
$A \to aab$
$B \to AC$
↳ non terminal

First convert all terminals to non terminals

$S \to ABX$
$A \to XXY$
$B \to AZ$
$X \to a$
$Y \to Bb$
$Z \to c$

Now, CNF a non-terminal can only generate two non-terminals.

$S \to \underline{AB}X$
$A \to \underline{XX}Y$ } 3 non terminals, create a pair of any two
$B \to AZ$
$X \to a$
$Y \to b$
$Z \to c$
$T \to AB$
$W \to XY$

Now generated :-

$S \rightarrow TX$
$A \rightarrow XW$
$B \rightarrow AZ$
$X \rightarrow a$
$Y \rightarrow b$
$Z \rightarrow c$
$W \rightarrow XY$
$T \rightarrow AB$

Example $S \rightarrow bA \mid aB$
$A \rightarrow bAA \mid aS \mid a$
$B \rightarrow aBB \mid bS \mid b$

$S \rightarrow XA \mid YB$
$A \rightarrow XAA \mid YS \mid a$
$B \rightarrow YBB \mid XS \mid b$
$X \rightarrow b$
$Y \rightarrow a$

$S \rightarrow XA \mid YB$
$A \rightarrow XZ \mid YS \mid a$
$B \rightarrow YW \mid XS \mid b$
$X \rightarrow b$
$Y \rightarrow a$
$Z \rightarrow AA$
$W \rightarrow YW$

Greibach Normal Form

GNF stands for greibach normal form. A CFG is in GNF (Greibach Normal Form) if all the production rules satisfy one of the following conditions.

→ A start symbol generation ε. For example, S → ε.
→ A non-terminal generation a terminal. For example, A → a.
→ A non-terminal generating a terminal which is followed by any numbers of non-terminals. For example, S → aASB

Example :-

S → AB
A → aA | bB | b
B → b

S → AB
Substitute A → aA | bB | b
S → aAB | bBB | bB
A → aA | bB | b
B → b