Unit-3

Context Free Grammars

3.1 Introduction to Context Free Grammars

Consider a given sentence she ran fast. The rules to form this sentence can be written as

sentence → <pronoun> <verb> <adverb>

pronoun → she

verb → ran

Adverb → fast

These rules are called production. Here, the words sentence, pronoun, verb and Adverb are called non-terminals or variables. Each production starts with non-terminal followed by an arrow followed by combination of non-terminals or terminals. The left hand side of the 1st production is called the start symbol. The set of rules to form a sentence which in ~~the~~ turn used to generate a language is called grammer.

A gramm G can be define as a 4 tuple or quadruple, $G = \{V, T, P, S\}$ where

V is the set of variables or non-terminals

T is set of terminals

P is set of rules called production

S is the start symbol.

In the above example sentence, pronoun, verb, adverb is set of non-terminals

or variables & she, ran, fastly is the set of terminals.

<sentence> → <pronoun><verb><Adverb> is production and sentence is the start symbol.

Each production is of the form $\alpha \rightarrow \beta$ where α is a non-empty string of terminals and/or non terminals and β is string of terminals and/or non-terminal including the null string i.e. α is string in $(VUT)^*$ and β is a string in $(VUT)^+$. In production, all the lower case letters, digits or any other special characteristics are considered as terminals and all the upper case letters are considered as non-terminals or variables. It should be noted that non-terminals can be replaced with string of terminals or non-terminals where as the terminals cannot be replaced or substitute

Production

→ Consider the production shown below:

$$A \rightarrow aA|a$$

Now, Consider, a string w of the form,

$$W = XAX$$

Here, the non-terminals A can be replaced

by 'aA' or 'a'. If A is replaced by aA, we have,

$$W = XaAX$$

This can be written as

$$XAX \Rightarrow XaAX$$

This means the XAX driven XaAX in on step i.e. by applying on production.

A production can be applied any number of times in an arbitary order.

In general can be written as,

$$W \Rightarrow W_1 \Rightarrow W_2 \Rightarrow \cdots \Rightarrow W_n$$

A string W_n is derived from and can be written as

$$W^* \Rightarrow W_n$$

Here, '*' indicates that unspecified number of production (including applying no production) unspeeified number of production have been applied to a get number of string W_n from W.

Also,

$$W^+ \Rightarrow W_n$$

Here, '+' indicates at least one or more product on applied to get the string W_n from W.

3.2 ✡ **Derivation :-**

Consider,

$$A \to \alpha B \gamma$$

and $B \to \beta$ are productions in Grammar G. Where, α, β and γ are string of terminals/non terminals and A and B are non terminals or variables. The non-terminals A directly derives the string $\alpha\beta\gamma$ by replacing the non terminal B in $\alpha B \gamma$ by the string β by applying the production $B \to \beta$ and can be written as

$$A \Rightarrow \alpha\beta\gamma$$

Thus the process of obtaining string terminal and/or non-terminals from the start symbol by applying some or all production is called derivations.

If a string is obtained by applying only one production, then it is called one step derivation and is denoted by $\Rightarrow$.

If one or more productions are applied to get a string $\alpha\beta\gamma$ from A, then we write.

$$A \overset{+}{\Rightarrow} \alpha\beta\gamma$$

If unspecified number of productions are applied (including applying no productions) to get string $\alpha\beta\gamma$ from A, then we write

$$A \overset{*}{\Rightarrow} \alpha\beta\gamma$$

Example:- Consider the grammar shown from which any arithmetic expression can be obtained:-

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

Obtain the string id+id*id-id from above grammar.

Ans→ $E \rightarrow E+E$
$\rightarrow id+E$ Applying $E \rightarrow id$
$\rightarrow id+E*E$ " $E \rightarrow E*E$
$\rightarrow id+id*E-E$ " $E \rightarrow id$ and $E \rightarrow E-E$
$\rightarrow id+id*id-E$ " $E \rightarrow id$
$\rightarrow id+id*id-id$ " $E \rightarrow id$

✡ Language:-

The given grammar can be generated the set of string consisting of only terminals by applying productions in different order. The set of such string is called language.

Let, $G=(V,T,P,S)$ be a grammar.
The language $L(G)$ generated by a grammar G is

$$L(G)=\{w \mid S \overset{*}{\Rightarrow} w \in T^*\}$$

HK-67

i.e., w is string of only terminals obtained from the start symbol S by applying an arbitary number of productions. The intermediate of terminals or/and non-terminals obtained during the derivation process is called sentential form of G.

Q.1. Let, $G=(V,T,P,S)$

where, $V=\{S,C\}$

$T=\{a,b\}$

S is the start symbol.

P is given as

$S \rightarrow aCa$

$C \rightarrow aCa \mid b$

Obtained the language generated by above gram

Soln

$S \rightarrow aCa$

$S \rightarrow aba$ (By applying $C \rightarrow b$)

Again,

$S \rightarrow aCa$

$\rightarrow aaCaa$ (By applying $C \rightarrow aCa$)

$\rightarrow aaaCaaa$

:

$\rightarrow aaa\cdots b\cdots aaa$ (By applying $C \rightarrow b$)

Therefore the language is generated by a given grammar is

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

HK-68

Q.2. Write a regular grammar that generates all set of palindromes over the $\Sigma = \{a, b\}$.

Soln → The recursive definition of palindrome is

(i) $\in$ or λ

(ii) a and b are palindromes

(iii) If w is palindromes, then string awa, bwb are palindromes.

Production corresponding to above definition are given as

$$S \to \in$$
$$S \to a \mid b$$
$$S \to aSa \mid bSb$$

Hence, the grammar, $G = (V, T, P, S)$ where,

V = non-terminal/variable. $V = \{S\}$

T = Terminal, $T = \{a, b\}$

P = Production

{

$S \to \in$

$S \to a \mid b$

$S \to aSa \mid bSb$

}

S = start symbol

Q.3. Obtain a Grammar to generate the langua

$$L = \{ 0^n 1^{n+1} \mid n \geq 0 \}$$

Soln→ The production to generate, $L = \{ 0^n 1^{n+1} \mid n$

is $A \rightarrow 0A1 \mid \epsilon$

To get the string $0^n 1^{n+1}$ and extract o should be placed. Production of this i given as

$$S \rightarrow A1$$

Hence, the grammar G to generate the language L, will be $G = (V, T, P, S)$ where,

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

$$T = \{0, 1\}$$

$$P = \{$$

$$S \rightarrow A1$$

$$A \rightarrow 0A1 \mid \epsilon$$

$$\}$$

where, S is the start symbol.

Q.4. Obtain the grammar to generate the langua

$$L = \{ w \mid n_a(w) = n_b(w) \}$$

Soln→ The production to generate the language $L = \{ w \mid n_a(b) = n_b(w) \}$ is

$$S \rightarrow aSb \mid bSa$$

To get the string $n_a(w) = n_b(w)$ i.e. n no. of equal to n no. of b. The production of thi

Q.3. Obtain a Grammar to generate the language
$L = \{0^n 1^{n+1} \mid n \geq 0\}$

Soln→ The production to generate, $L = \{0^n 1^{n+1} \mid n \geq 0\}$ is $A \rightarrow 0A1 \mid \epsilon$

To get the string $0^n 1^{n+1}$ and extract 01 is should be placed. Production of this is given as

$$S \rightarrow A1$$

Hence, the grammar G to generate the language L, will be $G = (V, T, P, S)$ where,

$V = \{S, A\}$

$T = \{0, 1\}$

$P = \{$

$S \rightarrow A1$

$A \rightarrow 0A1 \mid \epsilon$

$\}$.

where, S is the start symbol.

Q.4. Obtain the grammar to generate the languag

$$L = \{w \mid n_a(w) = n_b(w)\}$$

Soln→ The production to generate the language $L = \{w \mid n_a(b) = n_b(w)\}$ is

$$S \rightarrow aSb \mid bSa$$

To get the string $n_a(w) = n_b(w)$ i.e. n no. of a equal to n no. of b. The production of this

given as

$$S \rightarrow SS$$

Hence, the grammar G to generate the language L, will be

$$G = (V, T, P, S), \text{ where}$$

$$V = \{S\}$$

$$T = \{a, b\}$$

$$P = \{$$

$$S \rightarrow SS$$

$$S \rightarrow aSb \mid bSa$$

$$\}$$

where, S is the start symbol.

Q.5. Obtain grammar to obtain integer.

→ The sign of number can $+|-|\epsilon$. The production for this can be written as

$$S \rightarrow +|-|\epsilon$$

The production to obtained the digits of a number can be written as

$$D \rightarrow 0|1|2|\ldots\ldots|8|9$$

The production to obtained a number N can be written as,

$N \rightarrow D$ i.e. a no. N is a digit

$N \rightarrow DN$ i.e. the no. N followed by D also a number.

We know that an integer number can be a number N or sign of a number followed by number N. The production for this can be written as

$$I \rightarrow N | SN$$

Hence, the grammar G to obtain integer number can be written as,

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

where,

$$V = \{S, D, N, I\}$$

$$T = \{+, -, 0, 1, \cdots, 9\}$$

$$P = \{$$

$$I \rightarrow N | SN$$

$$N \rightarrow D$$

$$N \rightarrow DN$$

$$S \rightarrow + | - | \epsilon$$

$$D \rightarrow 0 | 1 | \cdots | 8 | 9$$

$$\}$$

where, I is the start symbol.

✡ Context Free Grammar (CFG)

The production in the regular grammar are restricted in two ways : ~~left~~ the left side must be a single variable i.e. non terminal while the right side have special form. The right hand side of the production

may contain 0 or more terminals. If non-terminals is present, that non-terminals should be left most symbol of the right most symbol ~~by~~ but, in a Context free Grammar there is only one restriction that is the symbol on the left hand side of the production should be a single non-terminal but there is no restriction on the right hand side of the production.

A Grammar $G=\{V,T,P,S\}$ is said to be Context free Grammar (CFG) or type2 grammar if all the productions are of the form.

$$A \rightarrow \alpha$$

where, $A \in V$ and
$\alpha \in (VUT)^*$

Here, the null string i.e. ϵ can appear on the R.H.S of the any production. There is only one symbol A on the left hand side of the production and that symbol must be non-terminals.

$\alpha \in (VUT)^*$ implies that R.H.S of the string may contain any no. of terminals and non-terminal including the null character. Every regular grammar is a Context free Grammar and hence a regular language is Context free language (CFL) but the reverse is not true always. Regular grammar is subset of Context

HK+73

free Grammar and the Regular language is subset of Context free language.

The language generated by Context free Grammar(CFG) is called Context free language(CFL). CFL is the most important aspect of formal language because it is applied to programming language. Actual programming languages have many features that can be described ~~gerst~~ elegently by means of Context free languages. What formal language theory tells has about context free languages, has important application in the design of programming languages as well as in the constructions of efficient compiler.

Q.1. Obtain CFG for the language

$$L=\{a^m b^n \mid m \neq n\}$$

Soln⇒ It is possible to construct a CFG for the given language. Let us assume that $m=n$. The production for this given that

$$S \longrightarrow aSb \mid \epsilon$$

Now, consider the following two cases for the given language.

HK-74.

case-I ($n>n$)

When no. of a's is more than that of no. of b's. The extra a's can be generated using the production.

$$A \rightarrow aA|a$$

This production append towards the left of the string generated from the production

$$S \rightarrow aSb|\epsilon$$

Then, we get production as,

$$S' \rightarrow AS$$

case-II($m<n$)

When ($m<n$) i.e. no of a's is less than that of no. of b's. The extra b's can be generated using the production

$$B \rightarrow bB|b$$

This production should be appended towards right of the string generated from the production

$$S \rightarrow aSb|\epsilon$$

Then, we get the production as,

$$S' \rightarrow SB$$

The context free Grammar $G=(V,T,P,S)$ where,

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

$T = \{a, b\}$

$P = \{$

$S' \rightarrow AS \mid SB$

$S \rightarrow aSb \mid \epsilon$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

$\}$

$S' \rightarrow SB$
$\rightarrow aSbB$
$\rightarrow abB$
$\rightarrow abbB$
$\rightarrow abbb$

Where, S' is the start Symbol.

Q.2. Obtain CFG for the regular expression $(011+1)^*(01)^*$

Soln → The production to generate 011+1 can be written as

$A \rightarrow 011 \mid 1$

$B \rightarrow \epsilon$

$B \rightarrow AB$

$A \rightarrow 011/1$

The regular expression $(011+1)^*$ may be an empty language or string consisting of any combination of 011's and 1's. The production for this can be written as

$B \rightarrow \epsilon$

$B \rightarrow AB$

$A \rightarrow 011 \mid 1 \mid \epsilon$

$S \rightarrow BD$

$B \rightarrow AB \mid \epsilon$

$A \rightarrow 011 \mid 1$

$D \rightarrow CD \mid \epsilon$

Now, the production to generate the string 01 can be written as

$C \rightarrow 01$

The production to generate the string $(01)^*$ can be written as

$$D \rightarrow \epsilon$$
$$D \rightarrow CD$$
$$C \rightarrow 01 \mid \epsilon$$

Now, the production to get the regular expression can be written as

$$S \rightarrow BD$$

Hence, the context free grammar for the regular expression is given by

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

where,

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

$T = \{0, 1\}$

$P = \{$

$S \rightarrow BD$

$B \rightarrow AB \mid \epsilon$

$A \rightarrow 011 \mid 1$

$D \rightarrow CD \mid \epsilon$

$C \rightarrow 01$

$\}$

where, S is the start symbol.

✡ Left most and Right most derivation:

A derivation is said to be left most if left most variable, i.e. left most non-terminal in the sentential form which replaced every steps. Derivation is said to be right most derivation if right most non-terminal in the sentential form is replaced at every steps:

$S \rightarrow BD$

$B \rightarrow AB \mid \epsilon$

$A \rightarrow 011 \mid 1$

$D \rightarrow CD \mid \epsilon$

$C \rightarrow 01$

✡ Derivation Tree (Parse Tree)

The left derivation or right derivation can be represented in the form of a tree such a tree is known as derivation tree or parse tree. The formal definition of Derivation Tree (or Parse Tree) is given as:

Let $G = (V, T, P, S)$ be a Context Free Grammar and an ordered is a tree for G if and only if it has a following property

(i) The root has the label S.

(ii) Every vertex has label which is in $(V \cup T \cup \epsilon)$

(iii) Every leaf node has label from terminal (T) and an interior vertex has a label from V.

(iv) If a vertex is labeled A and if $X_1, X_2, X_3, \ldots, X_n$ are all children of left, then P must contain production of the form

$$A \rightarrow X_1 X_2 X_3 X_4 \ldots X_n$$

Example:

The production of the grammar

$E \rightarrow (E)$

$E \rightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow id$

To obtain the expression (id * id) + id the derivation is given as,

$E \Rightarrow E+E$

$\Rightarrow (E)+E$

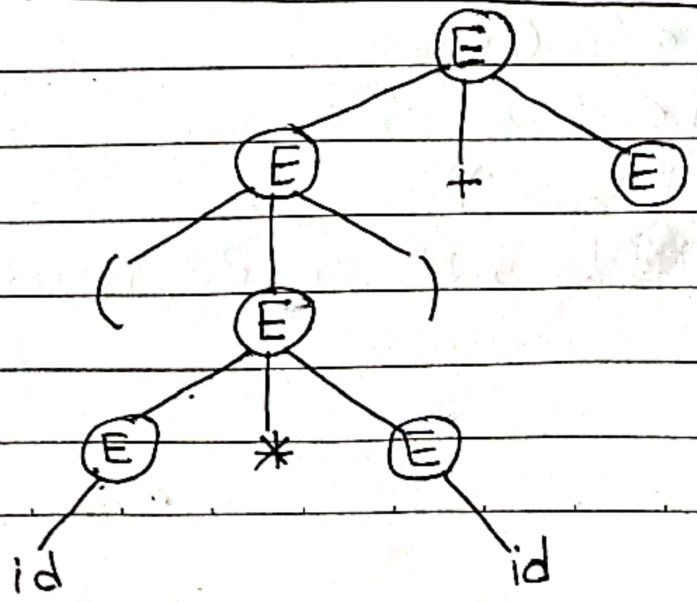
$\Rightarrow (E*E)+E$

$\Rightarrow (id*E)+E$

$\Rightarrow (id*id)+E$

$\Rightarrow (id*id)+id$

Left most derivation tree



✡ Partial Parse Tree:-

Let $G=(V,T,P,S)$ be CFG. The tree is partial derivation tree if it has the following properties:

(i) The root has the label S.

(ii) Every vertex has a label which is in (VUTUE)

(iii) Every leaf node has a label from (VUTUE). This property differs from that a derivation tree.

(iv) If a vertex is labeled A and if X1,X2,X3 are all children of left, then P must contain production of the form.

$$A \rightarrow X_1X_2X_3X_4$$

Example:- Consider the production of the given grammar as

$E \rightarrow (E)$

$E \rightarrow E+E$

$E \rightarrow E-E$

$E \rightarrow E*E$

$E \rightarrow E/E$

$E \rightarrow id$

If the left most partial derivation is given

$E \Rightarrow E+E$

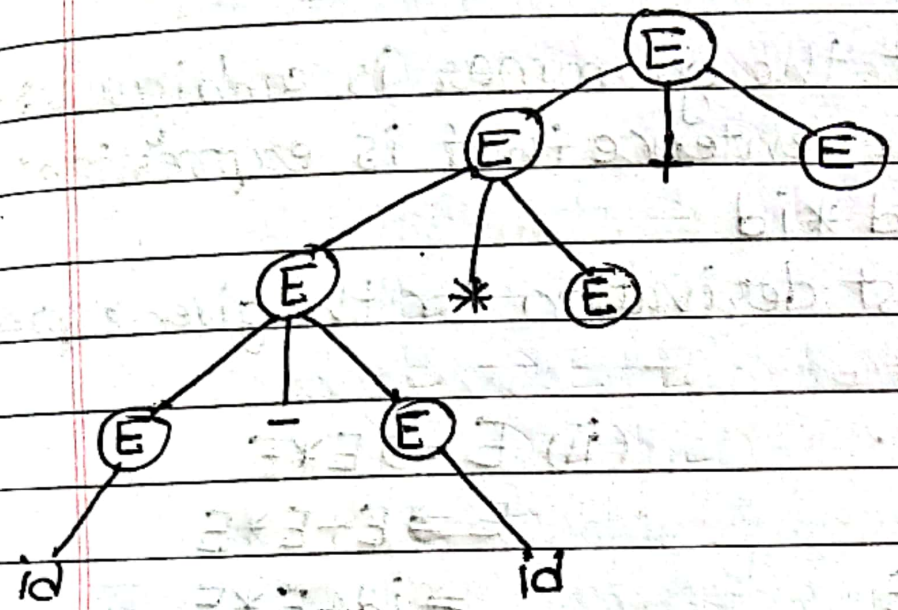
$\Rightarrow E*E+E$

$\Rightarrow E-E*E+E$

HK-80

$\Rightarrow$ id − E * E + E

$\Rightarrow$ id − id * E + E



✡ Ambiguous Grammar :-

A Grammar that has more than one leftmost derivation or more than one rightmost derivation for the same sentence is called ~~Ambigous~~ Ambiguous Grammar. This means that an Ambiguous grammar has more than one parse tree for the same sentence either by applying left most derivation twice or by applying rightmost derivation twice. Consider the production for the grammar for arithetic expression.

P = {

$E \rightarrow (E)$

$E \rightarrow E + E$

$E \rightarrow E - E$

$E \rightarrow E * E$

HC-8

$E \to E/E$

$E \to id$

4

Show that the grammar is ambiguous.

→ Consider the sentence that is expression.

id + id * id

The left most derivation for the given expression is given as

(i) $E \Rightarrow E+E$

$\Rightarrow id+E$

$\Rightarrow id+E*E$

$\Rightarrow id+id*E$

$\Rightarrow id+id*id$

(ii) $E \Rightarrow E*E$

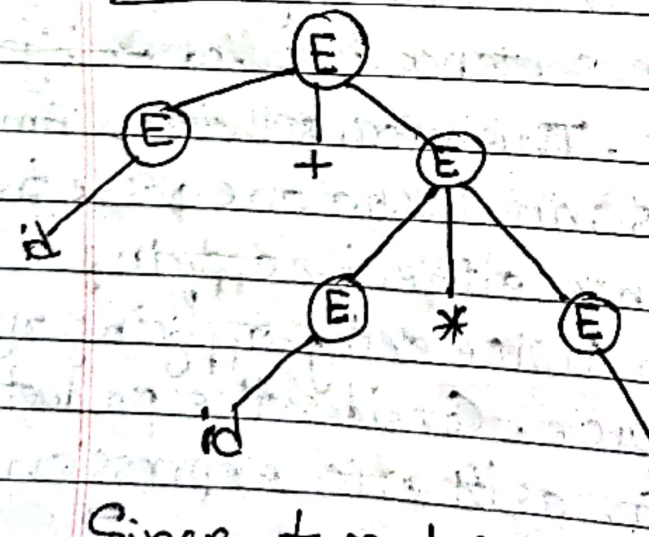
$E \Rightarrow E+E*E$

$\Rightarrow id+E*E$

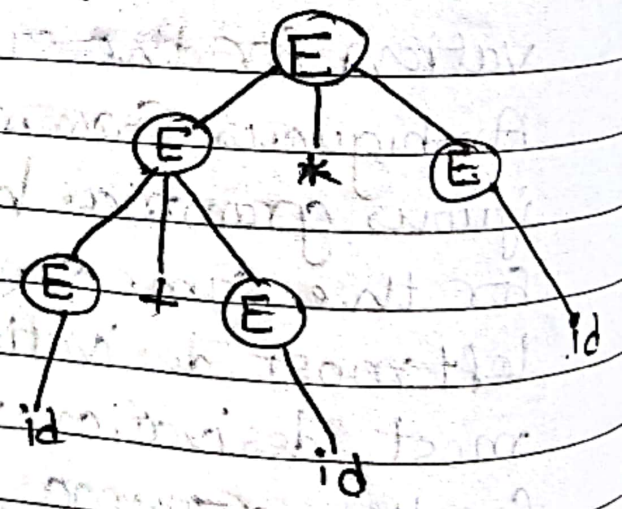
$\Rightarrow id+id*E$

$\Rightarrow id+id*id$

Tree (i)



Tree (ii)



Since two left most derivation trees are different for the same expression, the given grammar is ambiguous.

HK-82

✡ Parsing and Simple grammar i.e. S-Grammar.

Parsing is the method to check whether w can be derived from the grammar G. If we can be derived by applying leftmost derivation from the start symbol, then parsing is successful otherwise not. Parsing can be easily done by special type of Context Free Grammars called simple grammar or S-Grammar which can be formally defined as:

The simple grammar or S-Grammars is a special type of grammar where all productions are of the form

$A \rightarrow aX$. where, $A \in V$, $a \in T$ & $X \in V^*$ where, a pair (A, a) can occure at most once in p i.e., production i.e., if A is there on the left hand side of the production then there can be maximum of production where a is the 1st symbol on the right hand side of the production. Consider the grammar shows below which is S-Grammar. Where as the grammar

$S \rightarrow aABb$

$S \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

$A \rightarrow aA \mid a$

is not an S-Grammar because there are two production where the left hand side

of the production is S and the first symbol on the right hand side of the production is a.

✡ Determine the given grammar is ambiguous or not.

(i) $S \to aS|X$
$X \to aX|a$

Consider the two left most derivations for the string aaaa.

$S \Rightarrow aS$
$\Rightarrow aaS$
$\Rightarrow aaaS$
$\Rightarrow aaaX$
$\Rightarrow aaaa$

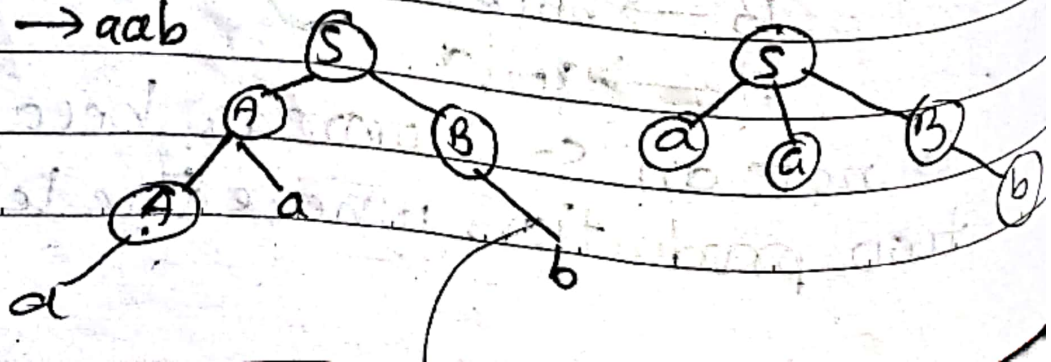
$S \Rightarrow X$
$\Rightarrow aX$
$\Rightarrow aaX$
$\Rightarrow aaaX$
$\Rightarrow aaaa$

Since, there are two left most derivations for the same sentence aaaa, the given grammar is ambiguous.

(ii) $S \to AB|aaB$
$A \to a|Aa$
$B \to b$

(a) $S \to AB$
$\to Aab$
$\to aaB$
$\to aab$

(b) $S \to aaB$
$\to aab$



HK-84

(iii) $S \longrightarrow aSbS$
$S \longrightarrow bSaS$
$S \longrightarrow \epsilon$

Simplification of Context Free Grammar (CFG)

The CFG imposes no restrictions whatsoever on the right side of the production. However, complete freedom is not necessary. There are several ways in which one can restrict the format of production without reducing the generative power of Context Free Grammar (CFG). In CFG, it may be necessary to eliminate some of the useless symbol and productions and we have the following method of simplifying the CFG.

(i) Elimination of useless variable or production
(ii) Elimination of ϵ or λ production.
(iii) Elimination of Unit production
(iv) Chomskey Normal Form (CNF)
(v) Greiback Normal Form (GNF)

1. Elimination of useless variable

Let $G=(V,T,P,S)$ be a context free Grammar. A variable $A \in V$ is said to be useful if and only if there is at least $w \in L(G)$. Such that

$$S \overset{*}{\Rightarrow} xAy \overset{*}{\Rightarrow} w$$

With x and y in $(V \cup T)^*$

HK-85

Any variable or symbol which is not reachable from the start symbol and which are not used while deriving a string are useless and all the productions which contain these symbol are useless production.

Eg.1. Eliminate useless symbols and productions from $G=(V,T,P,S)$ where,

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

$T=\{a,b\}$

$P=\{ S\to Bb$

$B\to bB|\epsilon$

$C\to ag$

$A\to ab\}$

Here, C and A are useless variables and the productions

$C\to aa$

$A\to ab$

are useless productions.

Eg.2. Eliminate useless symbols and productions from $G=(V,T,P,S)$ where

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

and $T=\{a,b\}$

with P consisting of

$S\to aS|A|C$

$A\to a$

$B\to aa$

$C\to aCb$

Soln⇒ Step1:- Identify the non-terminal which derives the terminal string.

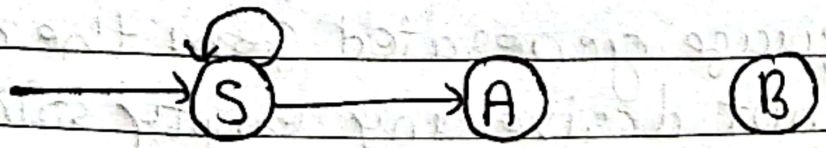
Here, A and B are variable that are non-terminals which derive the terminal strings. Therefore removing C from production we get production P_1 as:

$$P_1 = \{$$
$$S \rightarrow aS \mid A \mid BC$$
$$A \rightarrow a$$
$$B \rightarrow aa$$
$$\}$$

Step2: Identify the variables which are reachable from the starting states.

To identify this, draw the dependency graph from the set of productions i.e. P_1.

For the above productions the dependency graph is as follows.



Here, B is independent node and is not reached by the starting state. Eliminate that production. Then, P_1' becomes:

$$P_1' = \{$$
$$S \rightarrow aS \mid A \mid C$$
$$A \rightarrow a$$
$$\}$$

Hence, the simplified Grammar is

~~$G' = V', T', P'$~~

$$G = (V', T', P', S)$$

where,

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

$T' = \{a\}$

$P' = \{$

$S \rightarrow aS \mid A$

$A \rightarrow a$

$\}$

Here, S is the start symbol.

(2) Elimination of λ or ϵ production

If an empty string is not derived from the start symbol, then a production of the form $A \rightarrow \epsilon$ is not desirable. If the language generated from the grammar G doesnot derive any empty string and the grammar consists of ϵ or λ production then such ϵ production can be removed.

An production of a Context Free Grammar of the form $A \rightarrow \epsilon$ is called ϵ or λ production or null production. Any variable A for which the derivation $A \rightarrow \epsilon$ is possible is called nullable.

HK-88

Consider the production for the grammar

$S \rightarrow aCb$

$C \rightarrow aCb \mid \epsilon$

which generates ϵ-free language $a^n b^n \mid n \geq 1$, Here, ϵ production $C \rightarrow \epsilon$ can be eliminated after adding new production obtained by substituting ϵ for C. Then we get grammar as

$S \rightarrow aCb \mid ab$

$C \rightarrow aCb \mid ab$

Example:- Find a CFG without ϵ or λ production equivalent to the grammar defined by

$S \rightarrow ABCa \mid bD$

$A \rightarrow BC \mid b$

$B \rightarrow b \mid \epsilon$

$C \rightarrow c \mid \epsilon$

$D \rightarrow d$

Solution:- Step1:- Find all the nullable variables from the given production, we have

$B \rightarrow \epsilon$

$A \rightarrow BC$

$C \rightarrow \epsilon$

Here, the nullable variables are (B, C, A)

Step2:- Construct production without ϵ

Production P	Production P' without ϵ
$S \to ABCa$	$S \to ABCa \mid BCa \mid ACa \mid ABa \mid Ca \mid Ba \mid Aa \mid a$
$S \to bD$	$S \to bD$
$A \to BC$	$A \to BC \mid C \mid B$
$A \to b$	$A \to b$
$B \to b \mid \epsilon$	$B \to b$
$C \to c \mid \epsilon$	$C \to c$
$D \to d$	$D \to d$

Hence, the required grammar $G' = (V', T', P', S)$

Where,

$V' = \{A, B, C, D\}$

$T' = \{a, b, c, d\}$

$P' = \{$

$S \to ABCa \mid BCa \mid ABa \mid Ca \mid Ba \mid Aa \mid a$

$S \to bD$

$A \to BC \mid C \mid B$

$A \to b$

$B \to b$

$C \to c$

$D \to d$

$\}$

where, S is the start symbol.

HK-90

3. Elimination of unit production

Consider the production $A \to B$. The left hand side and Right hand side of the production contained only one variable. Such a production is called unit production. The formal definition of unit production is given as: Let $G=(V,T,P,S)$ be a Context Free Grammar. Any production in G of the form $A \to B$ where A and $B \in V$ is a unit production. Consider this production

$$A \longrightarrow B$$
$$B \longrightarrow bB \mid b$$

Where, $A \to B$ is unit production. Therefore, the production $A \to B$ can be eliminated and the resulting production can be written as

$$A \longrightarrow bB$$
$$B \longrightarrow bB \mid b$$

Q.1. Eliminate all unit production of the Grammar given below:

$$S \to AB$$
$$A \to a$$
$$B \to C \mid b$$
$$C \to D$$
$$D \to E \mid bC$$
$$E \to d \mid Ab$$

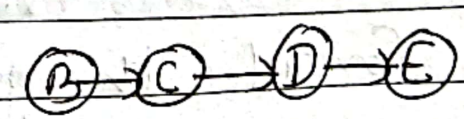


Fig: Dependency graph of unit production

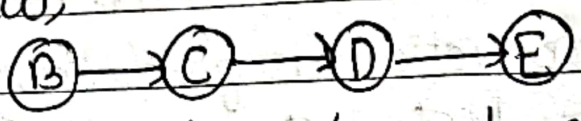
Solution :-

$B \rightarrow C$

$C \rightarrow D$

$D \rightarrow E$

The dependency graph of unit production is shown below,



It is clear from the dependency graph that all null unit production E can be generated from D. The null unit production from E are

$E \rightarrow d \mid Ab$

Since, $D \overset{*}{\Rightarrow} E$, we have

$D \rightarrow d \mid Ab \mid bC$

From dependency graph. It is clear that

$C \overset{*}{\Rightarrow} D$

So, null unit productions from D can be generated from C

$C \rightarrow d \mid Ab \mid bC$

Consider the production

$B \overset{*}{\Rightarrow} C$

So, null unit production

$B \rightarrow d \mid Ab \mid bC \mid b$

Hence, the final grammar after eliminating unit productions is given as

$G' = (V', T', P', S)$

where, $V' = \{S, A, B, C, D, E\}$

$T' = \{a, b\}$

$P' =$ {

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow d \mid Ab \mid bC \mid b$

$C \rightarrow d \mid Ab \mid bC$

$D \rightarrow d \mid Ab \mid bC$

$E \rightarrow d \mid Ab$

}

where, S is the start symbol.

✡ Normal Forms

In context free Grammar (CFG) the right hand side of a production can be any string of variables and terminals i.e. $(VUT)^*$ when production in G satisfy certain restictions, then G is said to be in normal form. The two normal forms are:

(i) Chomsky Normal Form (CNF)

(ii) Greibach Normal Form (GNF)

1. Chomsky Normal Form (CNF)

In CNF, R.H.S. of the production of a Grammar should ~~be~~ contain two or one symbols. If there are two symbols on the R.H.S. of the production, these two symbols must be non-terminal and if there is only one symbol that symbol must be a terminal.

CNF is defined as follows:
Let, $G=(V,T,P,S)$ be CFG. A grammar G is said to be CNF if all production are of the form:

$$A \to BC$$

or, $A \to a$ or, $S \to \epsilon$

where, A, B and $C \in V$
and $a \in T$

To convert the grammar into CNF, grammar should not have any ϵ production and unit production. If such production exist, we need to eliminate those production before converting a given grammar into CNF.

Q.1. Convert the grammar given below into CNF.

$S \rightarrow 0A \mid 1B$

$A \rightarrow 0AA \mid 1S \mid 1$

$B \rightarrow 1BB \mid 0S \mid 0$

Sol^n → Consider the productions which are not in CNF. Replace the terminal on the R.H.S. of the production by a non-terminal and introduce a new production. This step has to be carried out for each production which are not in CNF.

$A \rightarrow 1$
$B \rightarrow 0$ ---(i)

Given Production (P)	Action	Final Production (P')
$S \rightarrow 0A \mid 1B$	Replace 0 with B_0 i.e. $B_0 \rightarrow 0$ and 1 with B_1 $B_1 \rightarrow 1$	$S \rightarrow B_0A \mid B_1B$ $B_0 \rightarrow 0$ $B_1 \rightarrow 1$
$A \rightarrow 0AA \mid 1S$	Replace 0 by B_0 and 1 with B_1 $B_0 \rightarrow 0$ $B_1 \rightarrow 1$	$A \rightarrow B_0AA \mid B_1S$ $B_0 \rightarrow 0$ $B_1 \rightarrow 1$
$B \rightarrow 1BB \mid 0S$	$B_0 \rightarrow 0$ $B_1 \rightarrow 1$	$B \rightarrow B_1BB \mid B_0S$ $B_0 \rightarrow 0$ $B_1 \rightarrow 1$

Now, we need to restrict the no. of variables on R.H.S. of two production. The production obtained above which are in CNF are

$$S \to B_0A \mid B_1B$$
$$B_0 \to 0$$
$$B_1 \to 1$$
$$A \to B_1S$$
$$B \to B_0S$$

} ---- (ii)

The production which are not in CNF are

$$A \to B_0AA$$
$$B \to B_1BB$$

Considering $A \to B_0AA$

Replace AA on R.H.S. of production with variables D_0 i.e.

$$D_0 \to AA$$

So, resulting production

$$A \to B_0D_0$$
$$D_0 \to AA$$

} ---- (iii)

Similarly for $B \to B_1BB$

$$D_1 \to BB$$

So, resulting production is

$$B \to B_1D_1$$
$$D_1 \to BB$$

} ---- (iv)

Hence, the final grammar which is in CNF is given as $G' = (V', T', P', S)$

where,

$V' = \{S, B_0, A, B, B_1, D_0, D_1\}$

$T' = \{0, 1\}$

P' = Combing (ii), (iii) and (iv)

i.e. $P' = \{$

$S \rightarrow B_0A \mid B_1B$

$A \rightarrow B_1S \mid B_0D_0 \mid 1$

$B \rightarrow B_1S \mid B_1D_1 \mid 0$

$B_0 \rightarrow 0$

$B_1 \rightarrow 1$

$D_0 \rightarrow AA$

$D_1 \rightarrow BB$

$\}$

and S is the start symbol.

Example2 Convert the given CFG to CNF :

$G = \{S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \epsilon$

$\}$

Soln:- Note :- Steps for Converting CFG to CNF

Step1: If the start symbol S occurs on the right hand side of a grammar rule, create a new start symbol S' and a new production or

grammar rule $S' \rightarrow S$

step2:- Remove all production rules as well as unit production rules from the grammar.

step3:- Replace each production rule $A \rightarrow B_1B_2 \cdots B_n$ where $n>2$, with $A \rightarrow B_1C$ where $C \rightarrow B_2 \cdots B_n$.
Repeat this step for all production rules of the CFG having two or more symbols on the right hand side.

step4:- If the R.H.S. of any grammar rule is in the form $A \rightarrow \alpha B$ where 'α' is a terminal symbol and A, B are non-terminals, then the production rule is replaced by $A \rightarrow XB$ and $X \rightarrow a$.
Repeat this step for every production rule of the grammar which is of the form $A \rightarrow ab$

$$G = \{ S \rightarrow ASA \mid aB$$
$$A \rightarrow B \mid S$$
$$B \rightarrow b \mid \epsilon$$
$$\}$$

step1: If we observe the production rules we can find that in the first two rules starting state S appear R.H.S. of the productions. We need to add a new starting state S' and a new production rule $S' \rightarrow S$.

Step2:- Removing null production rules $B \to \epsilon$ & $A \to \epsilon$

After removing $B \to \epsilon$, grammar G.

$S' \to S$

$S \to ASA \mid aB \mid a$

$A \to B \mid S \mid \epsilon$

$B \to b$

After removing $A \to \epsilon$, grammar G.

$S' \to S$

$S \to ASA \mid aB \mid a \mid AS \mid SA$

$A \to B \mid S$

$B \to b$

Removing unit production rules, $S' \to S$, $A \to B$ & $A \to S$

After removing $S' \to S$, grammar G.

$S' \to ASA \mid aB \mid a \mid AS \mid SA$

$S \to ASA \mid aB \mid a \mid AS \mid SA$

$A \to B \mid S$

$B \to b$

After removing $A \to B$, grammar G

$S' \to ASA \mid aB \mid a \mid AS \mid SA$

$S \to ASA \mid aB \mid a \mid AS \mid SA$

$A \to b \mid S$

$B \to b$

After removing $A \to S$, grammar G

$S' \to ASA \mid aB \mid a \mid AS \mid SA$

$S \to ASA \mid aB \mid a \mid AS \mid SA$

$A \to b \mid ASA \mid aB \mid a \mid AS \mid SA$

$B \to b$

HK-99

Step3:- We find some production rules that have more than two symbols on the R.H.S.

$S' \rightarrow ASA$

$S \rightarrow ASA$

$A \rightarrow SA$

Let's replace SA by X i.e. $X \rightarrow SA$

~~$S' \Rightarrow AX|aB|$~~

$S' \rightarrow AX|aB|a|AS|SA$

$S \rightarrow AX|aB|a|AS|SA$

$A \rightarrow b|AX|aB|a|AS|SA$

$B \rightarrow b$

$X \rightarrow SA$

Now we will replace terminal symbol a by Y from these rules, $S' \rightarrow aB$, $S \rightarrow aB$ and $A \rightarrow aB$.

Now, the grammar G in CNF

$S' \rightarrow AX|YB|a|AS|SA$

$S \rightarrow AX|YB|a|AS|SA$

$A \rightarrow b|AX|YB|a|AS|SA$

$B \rightarrow b$

$X \rightarrow SA$

$Y \rightarrow a$

Advantages of CNF

- Simple and standardized grammar
- Efficient parsing Algorithm

- less Grammar Size
- Elimination of Empty string Rules

Disadvantages of CNF

- Increased Size
- Loss of Readability
- Complexity of Transformation

H/W.

Q.1. Convert the following CFG into CNF

(i) $S \rightarrow ASA \mid aB$
$A \rightarrow B \mid S$
$B \rightarrow b \mid \epsilon$

(ii) $S \rightarrow aA$
$A \rightarrow a$
$B \rightarrow c$

(iii) $S \rightarrow a \mid aA \mid B$
$A \rightarrow aBB \mid \epsilon$
$B \rightarrow Aa \mid b$

(iv) $S \rightarrow aXX$
$X \rightarrow aS \mid bS \mid a$

2. Greibach Normal Forms (GNF)

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

$S \rightarrow \epsilon$

$A \rightarrow a$

$S \rightarrow a\alpha$

where, $\alpha \in V^*$

Example

$S \rightarrow aAB \mid aB$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

2. Greibach Normal Forms (GNF)

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

$S \rightarrow \epsilon$

$A \rightarrow a$

$S \rightarrow a\alpha$

where $\alpha \in V^*$

Example

$S \rightarrow aAB \mid aB$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

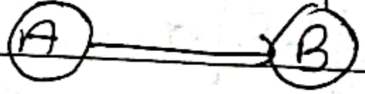
Q1. Convert the given grammar into GNF:

$S \rightarrow AB1 \mid 0$

$A \rightarrow 00A \mid B$

$B \rightarrow 1A1$

Ans→ Since the given grammar doesnot have any ϵ production, so we now eliminate unit production $A \rightarrow B$. The dependency graph is as



It is clear from dependency graph that A→B. So, all the symbols derivable from B are also derivable from A. So in the production

$A \to 00A \mid B$, the variable B can be replaced by the string 1A1 using the production $B \to 1A1$. Then the given grammar becomes

$S \to AB1 \mid 0$

$A \to 00A \mid 1A1$

$B \to 1A1$

Next convert the above grammar into CNF, first replace the terminal on the RHS of the production which is not in CNF with variable. Here, replace 1 with A_1 and 0 with A_0 in productions which are not in CNF. Now the grammar becomes

$S \to ABA_1 \mid 0$

$A \to A_0A_0A \mid A_1AA_1$

$B \to A_1AA_1$

$A_0 \to 0$ $\quad A_1 \to 1$

$A_1 \to 1$ $\quad A_0 \to 0$

Now, the restrict the number of variables. In RHS of production to two. Then the resulting grammar becomes

$S \to AD_1 \mid 0$

$A \to A_0D_2 \mid A_1D_3$

$B \to A_1D_3$

$A_1 \to 1$

$A_0 \to 0$

$D_1 \to BA_1$

$D_2 \to A_0A$

$D_3 \to AA$

HK103

Now, Converting this grammar which is in CNF into GNF. Rename all the variables as shown below

let: $S \Rightarrow A_1$

$A \Rightarrow A_2$

$B \Rightarrow A_3$

$A_1 \Rightarrow A_4$

$A_0 \Rightarrow A_5$

$D_1 \Rightarrow A_6$

$D_2 \Rightarrow A_7$

$D_3 \Rightarrow A_8$

The given production becomes

$A_1 \longrightarrow A_2A_6 | 0$

$A_2 \longrightarrow A_5A_7 | A_4A_8$

$A_3 \longrightarrow A_4A_8$

$A_4 \longrightarrow 1$

$A_5 \longrightarrow 0$

$A_6 \longrightarrow A_3A_4$

$A_7 \longrightarrow A_5A_2$

$A_8 \longrightarrow A_2A_4$

Consider production $A_3 \longrightarrow A_4A_8$

By using the production $A_4 \longrightarrow 1$, we get

$A_3 \longrightarrow 1A_8$, which is in GNF.

Consider production: $A_6 \longrightarrow A_3A_4$

Using $A_3 \longrightarrow 1A_8$, we get

HK104

$A_6 \rightarrow 1A_8A_4$, which is in GNF.

Consider: $A_2 \rightarrow A_5A_7 | A_4A_8$

Using $A_5 \rightarrow 0$, $A_4 \rightarrow 1$

$A_2 \rightarrow 0A_7 | 1A_8$, which is in GNF.

Consider: $A_1 \rightarrow A_2A_6$

Using $A_2 \rightarrow 0A_7 | 1A_8$, we get

$A_1 \rightarrow (0A_7 | 1A_8)A_6$

$A_1 \rightarrow 0A_7A_6 | 1A_8A_6$ which is in GNF

Consider: $A_7 \rightarrow A_5A_2$

Using $A_5 \rightarrow 0$, we get

$A_7 \rightarrow 0A_2$ which is in GNF.

Consider: $A_8 \rightarrow A_2A_4$

Using $A_2 \rightarrow 0A_7 | 1A_8$, we get

$A_8 \rightarrow (0A_7 | 1A_8)A_4$

$A_8 \rightarrow 0A_7A_4 | 1A_8A_4$

Finally, the converted grammar in GNF be

$G' = (V', T', P', S)$

$T' = \{0, 1\}$

$P' = \{$

$A_1 \rightarrow 0A_7A_6 | 1A_8A_6$

$A_2 \rightarrow 0A_7 | 1A_8$

$A_3 \rightarrow 1A_8$

HKIDS

$A_4 \rightarrow 1$

$A_5 \rightarrow 0$

$A_6 \rightarrow 1A_8A_4$

$A_7 \rightarrow 0A_2$

$A_8 \rightarrow 0A_7A_4 \mid 1A_8A_4$

Y

Where A_1 i.e. S is the start symbol.

Q.2. Convert the following grammar

$A \rightarrow BC$

$B \rightarrow CA \mid b$

$C \rightarrow AB \mid a$

into GNF

HK106