4 CONTEXT FREE GRAMMAR

4.1 Objective

After studying this chapter, students will be able to

- Define Context Free Grammar (CFG)
- Define Derivation Tree
- Find ambiguity in the context free grammar
- Simplify the context free grammar
- Eliminate the Null-Production from CFG
- Eliminate the Unit-Production from CFG
- Define various Normal forms
- Convert CFG into Chomsky Normal Form (CNF)
- Convert CFG into Greibach Normal Form (GNF)
- Know Pumping Lemma for context free grammar

4.2 Introduction

Context free grammars (CFGs) have important application and role in defining the syntactic structure of the programming languages and compiler construction.

We shall study about how to represent the string derived by a CFG in the form of derivation tree (or called **Parse Tree**).

Existance of the grammar which may derive a given string in more than one ways makes the grammar ambiguous. So, if the grammar is ambiguous, the question arises that can we remove the ambiguity from the grammar? The answer to such questions lies in this section.

Next, we shall study the various canonical forms of the CFGs, namely, Chomsky Normal Forms and Greibach Normal Forms.

Next Section focuses on the languages which are not defined by a context free grammar by using pumping lemma.

4.3 Context Free Grammars

In the preceeding chapters, we discussed the class of regular languages, but these regular languages are not effective in describing a simple kind of nested structure or for balancing in programming languages parenthesis so we required a wider class of languages to cover more complicated features.

The class of context free languages (grammars) is the most important class of languages as they are applied in number of applications such as **programming languages, parser design, lexical analysis** of compilers, **string matching algorithm** and formats used for information exchange.

4.3.1 Definition

Now, we give formal definition of the grammar.

A GRAMMAR $G = \{V_N, \Sigma, P, S\}$ is said to be **context free grammar** or **type-2** grammar if each production $\alpha \rightarrow \beta$ satisfies the following condition in addition to being type-1 grammar.

$$|\alpha| = 1 \text{ where,}$$

$\alpha \in V_N$ and $\beta \in (V_N \cup \Sigma)^*$

4.4 Derivation Tree (Parse Tree)

A '**Derivation Tree**' is an ordered tree in which the nodes are labelled with the left sides of the production and children of the node represent its corrosponding right side.

4.4.1 Definition

Let $G = \{V_N, \Sigma, P, S\}$ be a CFG. An ordered tree is a derivation tree for G if it has following **proporties.**

(i) There is a node, known as root of the tree.

(ii) Each and every leaf in the tree has a label from $(\Sigma \cup \wedge)$.

(iii) Each and every interior vertex (a vertex which has no leaf) has a label from V_N.

(iv) If a vertex has label $A \in V_N$, and its children are labelled (from left to right) a_1, a_2, a_3, ... a_n, then P must contain a production of the form $A \rightarrow a_1 a_2 a_3 .... a_n$.

(v) A leaf labelled $\wedge$ has no child.

For example, the sample derivation tree is shown in Fig. 4.1.

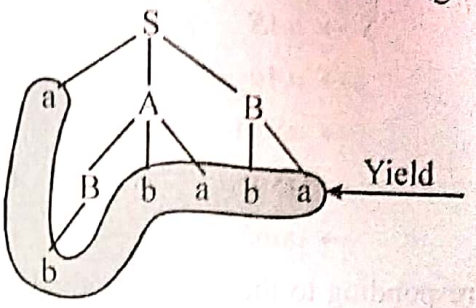


Fig. 4.1 : Derivation Tree.

4.4.2 Yield of The Derivation Tree

The yield of the derivation tree is the **concatenation of the labels of the leaves** without repetition in the left to right ordering.

The yield of the derivation tree of Fig. 4.1 is ***abbaba.***

Example 4.1 : Consider the grammar G whose productions are

$$S \rightarrow aAS \ / \ a$$

$$A \rightarrow SbA \ / \ SS \ / \ ba.$$

Show that *S* derives *aabbaa* and construct the derivation tree whose yield is *aabbaa*.

Solution : Given : $S \rightarrow aAS / a$

$A \rightarrow SbA / SS / ba$

$S \rightarrow aAS$

$\rightarrow aSbAS$ $(A \rightarrow SbA)$

$\rightarrow aabAS$ $(S \rightarrow a)$

$\rightarrow aabbaS$ $(A \rightarrow ba)$

$\rightarrow aabbaa$ $(S \rightarrow a)$ ....(4.1)

Hence, S derives aabbaa.

The derivation tree is given below :

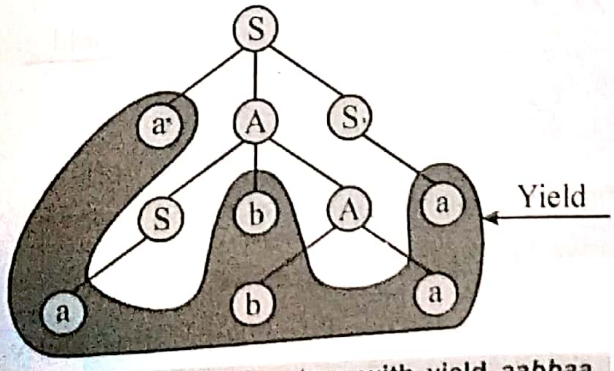


Fig. 4.2 : Derivation tree with yield aabbaa.

Another derivation for ***aabbaa*** is :

$$
\begin{aligned}
S &\rightarrow aAS \\
&\rightarrow aAa && (S \rightarrow a) \\
&\rightarrow aSbAa && (A \rightarrow SbA) \\
&\rightarrow aSbbaa && (A \rightarrow ba) \\
&\rightarrow aabbaa && (S \rightarrow a)
\end{aligned} \qquad \ldots(4.2)
$$

The **derivation tree** corresponding to the above productions is :

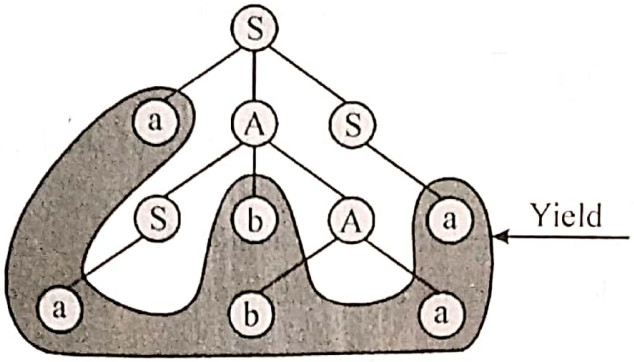


Fig. 4.3 : Derivation tree with yield *aabbaa*.

One more derivation for the string **aabbaa** is :

$$
\begin{aligned}
S &\rightarrow aAS \\
&\rightarrow aSbAS && (A \rightarrow SbA) \\
&\rightarrow aSbAa && (S \rightarrow a) \\
&\rightarrow aabAa && (S \rightarrow a) \\
&\rightarrow aabbaa && (A \rightarrow ba)
\end{aligned} \qquad \ldots(4.3)
$$

The **derivation tree** for the productions is :

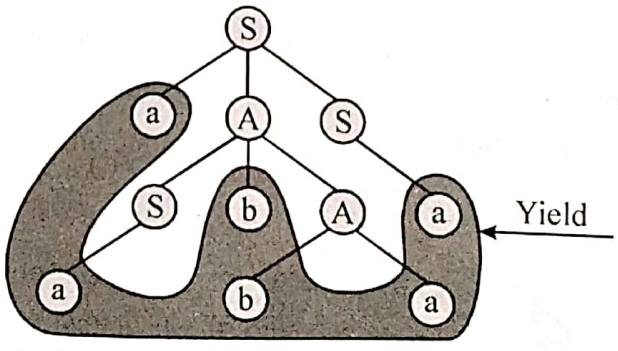


Fig. 4.4 : Derivation tree with yield *aabbaa*.

4.4.3 Left Most Derivation

A derivation $A \xrightarrow{*} w$ is called a left most derivation, if we apply a production only to the leftmost variable / non-terminal at every step.

As in equation (4.1), the derivation is leftmost derivation.

4.4.4 Rightmost Derivations

A derivation $A \xrightarrow{*} w$ is a rightmost derivation if we apply production to the right most variable/non-terminal at every step.

As in equation (4.2), the derivation is rightmost derivation.

Note : Some derivation are neither left most derivation nor rightmost derivations.

Equation (4.3) is an example of such derivation.

Example 4.2 : Consider the grammar G with set of production rules P given below:

$$S \rightarrow 0B / 1A$$

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

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

Find :

(a) Leftmost derivation

(b) Rightmost derivation

(c) Derivation Tree

For the string $W = 00110101$.

Solution :

(a) Leftmost derivation :

$$\begin{aligned} S &\rightarrow 0B \\ &\rightarrow 00BB && (B \rightarrow 0BB) \\ &\rightarrow 001B && (B \rightarrow 1) \\ &\rightarrow 0011S && (B \rightarrow 1S) \\ &\rightarrow 00110B && (S \rightarrow 0B) \\ &\rightarrow 001101S && (B \rightarrow 1S) \\ &\rightarrow 0011010B && (S \rightarrow 0B) \\ &\rightarrow 00110101 && (B \rightarrow 1) \end{aligned} \quad \text{....(4.4)}$$

(b) Rightmost Derivation :

$$\begin{aligned} S &\rightarrow 0B \\ &\rightarrow 00BB && (B \rightarrow 0BB) \\ &\rightarrow 00B1S && (B \rightarrow 1S) \\ &\rightarrow 00B10B && (S \rightarrow 0B) \\ &\rightarrow 00B101S && (B \rightarrow 1S) \end{aligned}$$

$\rightarrow 00B1010B$ $(S \rightarrow 0B)$

$\rightarrow 00B10101$ $(B \rightarrow 1)$

$\rightarrow 00110101$ $(B \rightarrow 1)$

(c) Derivation Tree : ....(5.5)

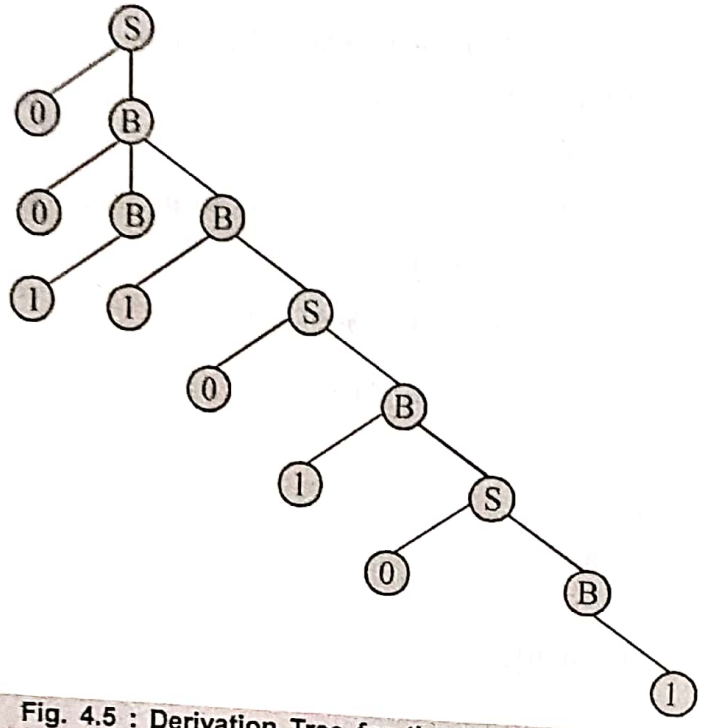


Fig. 4.5 : Derivation Tree for the string 00110101.

4.4.5 Ambiguity in Context Free Grammar (CFG)

Grammars can be used to put structure on programs and documents. The assumption was that a grammar uniquely determines a structure for each string in its language. However, not every grammar does provide unique structure.

For example, consider the following sentence in English "In books selected information is given." The word "**Selected**" may refer to books or informatioin so there are two different interpretations of the same sentence.

When a grammar fails to provide unique structure, it is sometimes possible to redesign the grammar to make the structure unique for each string in the language. Unfortunately, sometimes we can not do so, i.e., they are inherently ambiguous.

A Context Free Grammar (CFG) is called **ambiguous**, if there exists more than one parse tree or derivation tree for some string w.

A Context Free Grammar (CFG) is called **unambiguous** if there is one and only one parse tree or derivation trees for the given string.

A Context Free Grammar (CFG) is called **inherently ambiguous** if the grammar can not be converted into unambiguous grammar.

Example 4.3 : Consider the Context Free Grammar (CFG) G whose productions are $S \rightarrow SbS/a$. Show that the grammar is ambiguous.

Solution : To prove that the given Grammar G is ambiguous. Let us consider the string w = abababa We can derive this string by applying production rules as follows :

$$
\begin{aligned}
S &\rightarrow SbS \\
&\rightarrow abS && (S \rightarrow a) \\
&\rightarrow abSbS && (S \rightarrow SbS) \\
&\rightarrow ababS && (S \rightarrow a) \\
&\rightarrow ababSbS && (S \rightarrow SbS) \\
&\rightarrow abababS && (S \rightarrow a) \\
&\rightarrow abababa && (S \rightarrow a)
\end{aligned}
$$

The derivation tree for the string w = abababa by using above productions is shown in Fig. 4.6.

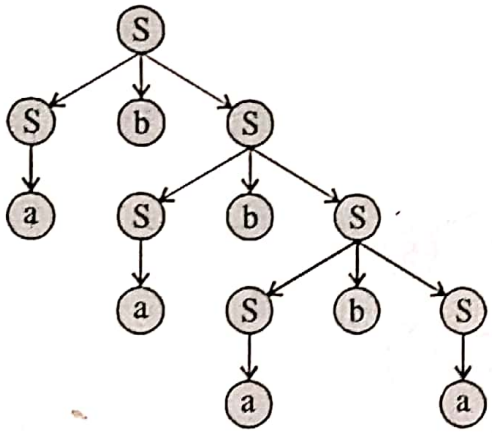


Fig. 4.6

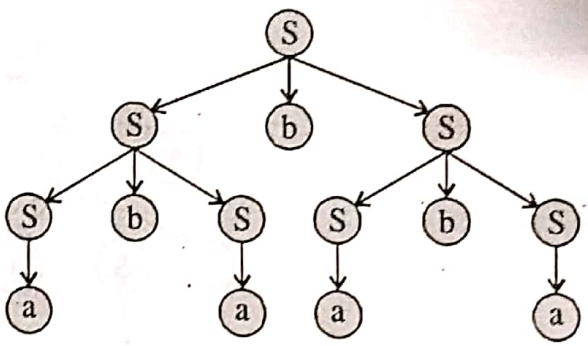


Fig. 4.7

We can derive this string by applying the productions in this way also.

$$
\begin{aligned}
S &\rightarrow SbS \\
&\rightarrow SbSbS && (S \rightarrow SbS) \\
&\rightarrow abSbS && (S \rightarrow a) \\
&\rightarrow ababS && (S \rightarrow a) \\
&\rightarrow ababSbS && (S \rightarrow SbS) \\
&\rightarrow abababS && (S \rightarrow a) \\
&\rightarrow abababa && (S \rightarrow a)
\end{aligned}
$$

The derivation tree corrosponding to the above production is shown in Fig. 4.7, so there exists more than one derivation trees for the string hence, the grammar is ambiguous.

Example 4.4 : Show that the following *CFG* is ambiguous.

$$S \rightarrow aSbS \mid bSaS \mid \wedge$$

Solution :

Given : $S \rightarrow aSbS \mid bSaS \mid \wedge$

To prove : Grammar is ambiguous.

Let us consider the string $w = abab$. We can derive the string by applying the productions as follows :

$$\begin{aligned} S &\rightarrow a\mathbf{S}bS \\ &\rightarrow ab\mathbf{S}aSbS && (S \rightarrow bSaS) \\ &\rightarrow aba\mathbf{S}bS && (S \rightarrow \wedge) \\ &\rightarrow abab\mathbf{S} && (S \rightarrow \wedge) \\ &\rightarrow abab && (S \rightarrow \wedge) \end{aligned}$$

The corrosponding, derivation tree is :

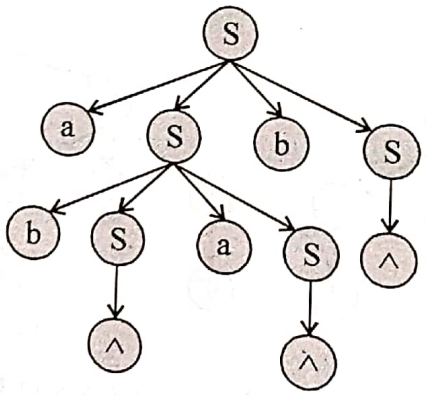


Fig. 4.8 : Derivation tree.

We can also derive the string $w = abab$ by applying the productions in this way.

$$\begin{aligned} S &\rightarrow aSb\mathbf{S} \\ &\rightarrow a\mathbf{S}baSbS && (S \rightarrow aSbS) \\ &\rightarrow aba\mathbf{S}bS && (S \rightarrow \wedge) \\ &\rightarrow abab\mathbf{S} && (S \rightarrow \wedge) \\ &\rightarrow abab && (S \rightarrow \wedge) \end{aligned}$$

The derivation tree by using above production is :

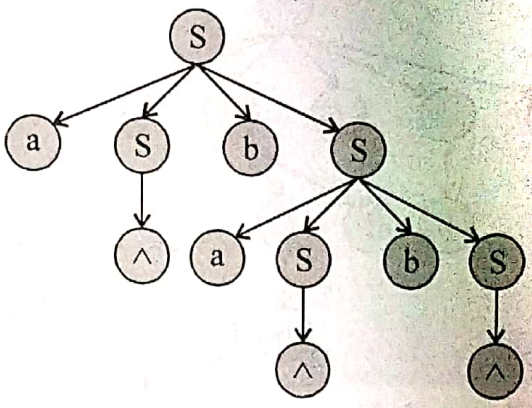


Fig. 4.9 : Derivation tree.

Since, there are more than one deviation tree for the string, w = abab, hence, the grammar is **ambiguous**.

Example 4.5 : Show that the grammar given below is ambiguous.

$$S \rightarrow iCtS$$
$$S \rightarrow iCtSeS$$
$$S \rightarrow a$$
$$C \rightarrow b$$

Solution : Given :

$$S \rightarrow iCtS$$
$$S \rightarrow iCtSeS$$
$$S \rightarrow a$$
$$C \rightarrow b$$

To prove : The grammar is ambiguous.

Let us consider the string $w = ibtibtibtaea$

We can derive the string $w = ibtibtibtaea$ as follows:

(i)

$$S \rightarrow iCtSeS$$
$$\rightarrow ibtSeS \qquad (C \rightarrow b)$$
$$\rightarrow ibtiCtSes \qquad (S \rightarrow iCtS)$$
$$\rightarrow ibtibtSeS \qquad (C \rightarrow b)$$
$$\rightarrow ibtibtiCtSeS \qquad (S \rightarrow iCtS)$$
$$\rightarrow ibtibtibtSeS \qquad (C \rightarrow b)$$
$$\rightarrow ibtibtibtaeS \qquad (S \rightarrow a)$$
$$\rightarrow ibtibtibtaea \qquad (S \rightarrow a)$$

Derivation tree for the above productions :

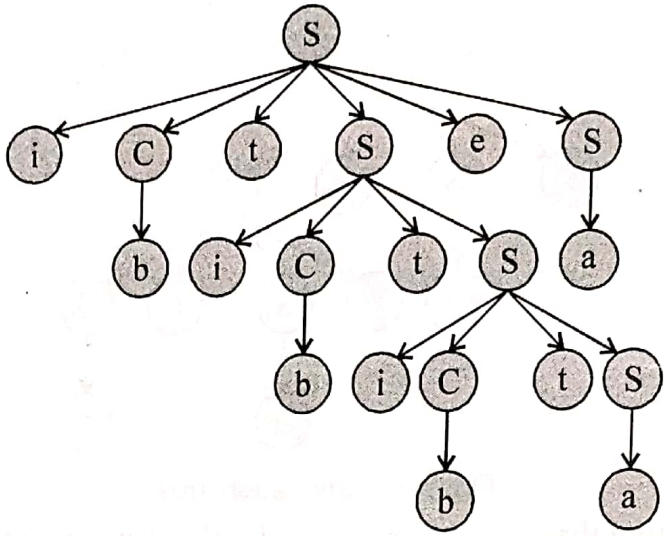


Fig. 4.10 : Derivation tree.

(ii)

$S \rightarrow iCtS$

$\rightarrow ibtS$ $(C \rightarrow b)$

$\rightarrow ibtiCtS$ $(S \rightarrow iCtS)$

$\rightarrow ibtibtS$ $(C \rightarrow b)$

$\rightarrow ibtibtiCtSeS$ $(S \rightarrow iCtSeS)$

$\rightarrow ibtibtibtSeS$ $(C \rightarrow b)$

$\rightarrow ibtibtibtaeS$ $(S \rightarrow a)$

$\rightarrow ibtibtibtaea$ $(S \rightarrow a)$

The derivation tree

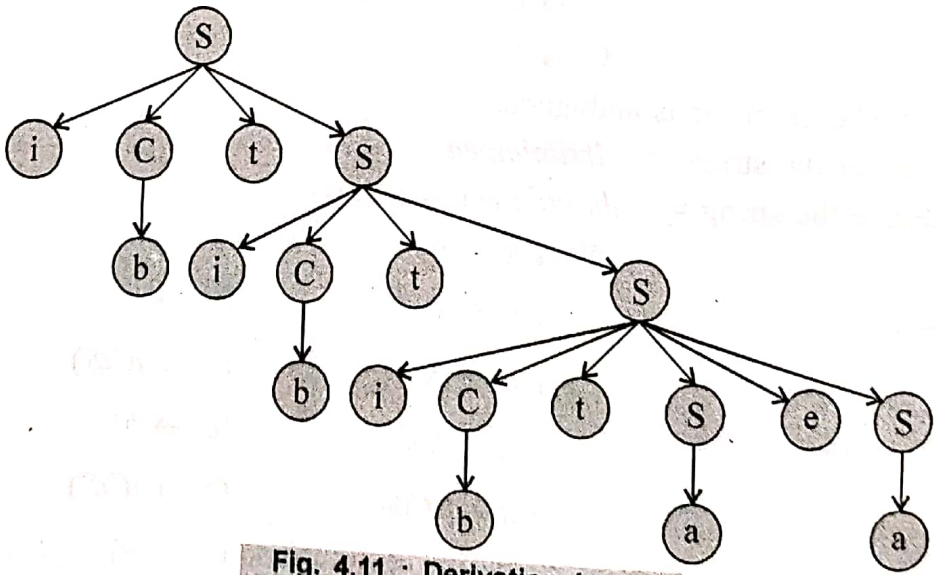


Fig. 4.11 : Derivation tree.

Since, there exists more than one derivation tree for the string w = ***ibtibtibtaea*** hence, the grammar is ambiguous.

Example 4.6 : Check whether the grammar $G = (V_N, \Sigma, P, S)$ is ambiguous or not.

where,

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

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

$$P = \{S \rightarrow AA$$

$$A \rightarrow AAA$$

$$A \rightarrow a$$

$$A \rightarrow bA$$

$$A \rightarrow Ab\}$$

Solution : Let us consider the string $w = babbab$

Let us derive the string $w = babbab$ using above production rules:

(i)

$$S \rightarrow AA$$

$$\rightarrow AbA \qquad (A \rightarrow Ab)$$

$$\rightarrow bAbA \qquad (A \rightarrow bA)$$

$$\rightarrow babA \qquad (A \rightarrow a)$$

$$\rightarrow babbA \qquad (A \rightarrow bA)$$

$$\rightarrow babbAb \qquad (A \rightarrow Ab)$$

$$\rightarrow babbab \qquad (A \rightarrow a)$$

Derivation tree for the above production is :

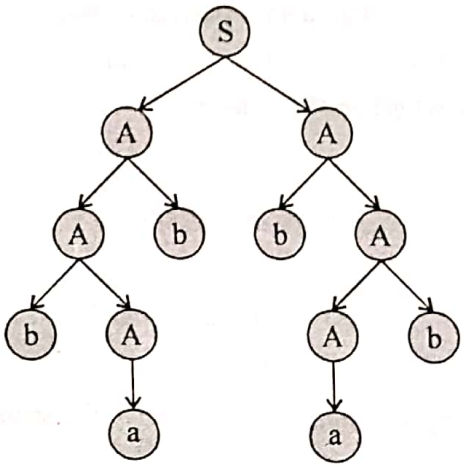


Fig. 4.12 : Derivation tree.

(ii)
$$S \rightarrow AA$$
$$\rightarrow bAA \qquad (A \rightarrow bA)$$
$$\rightarrow baA \qquad (A \rightarrow a)$$
$$\rightarrow baAb \qquad (A \rightarrow Ab)$$
$$\rightarrow babAb \qquad (A \rightarrow bA)$$
$$\rightarrow babbAb \qquad (A \rightarrow bA)$$
$$\rightarrow babbab \qquad (A \rightarrow b)$$

Derivation tree for the above productions is :

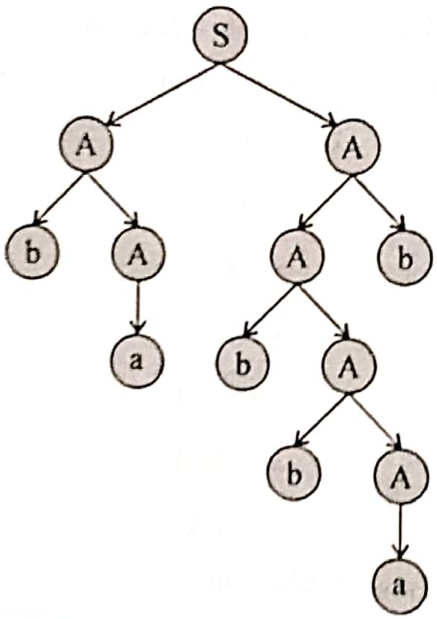


Fig. 4.13 : Derivation tree.

Since, there exits more than one derivation tree for the string hence the grammar is **ambiguous.**

Example 4.7 : Check whether the grammar

$$S \rightarrow a \mid abSb \mid aAb$$
$$A \rightarrow bS \mid aAAb$$

is ambiguous or not.

Solution : Given :

$$S \rightarrow a \mid abSb \mid aAb$$
$$A \rightarrow bS \mid aAAb$$

Let us consider the string $w = ababbb$ and derive the string from above production rules,

(i)
$$S \rightarrow abSb$$
$$\rightarrow abaAbb \qquad (S \rightarrow aAb)$$

$\rightarrow ababSbb$ $(A \rightarrow bS)$

$\rightarrow abababb$ $(S \rightarrow a)$

Derivation tree for the production rules :

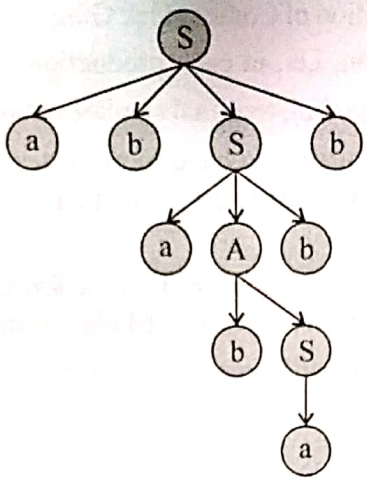


Fig. 4.14 : Derivation tree.

(ii) The string $w = ababbb$ can also be derived as :

$S \rightarrow aAb$

$\rightarrow abSb$ $(A \rightarrow bS)$

$\rightarrow ababSbb$ $(S \rightarrow abSb)$

$\rightarrow abababb$ $(S \rightarrow a)$

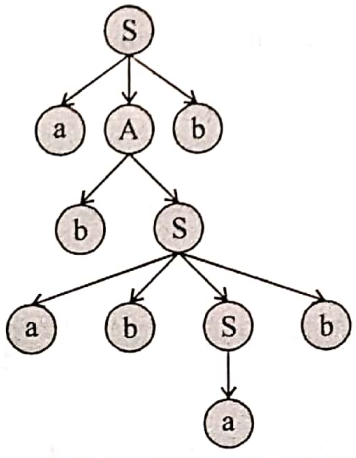


Fig. 4.15 : Derivation tree.

Since, there exists more than one derivation tree for $w = ababbb$, hence, the grammar is **ambiguous.**

4.5 Simplifications of CFG

4.5.1 Introduction

As it is clear from the definition of Context Free Grammar (CFG) that it imposes no restriction on the right side of the production, i.e., in each production $\alpha \rightarrow \beta$, where $\beta \in (V_N \cup \Sigma)^*$ may contain variable (Non-terminal) symbol, terminal symbol or even Λ (null).

In many instances, it is desirable to have more restrictions on the grammar. Due to this, we need to look at method for transforming the context free grammar into an equivalent grammar that satisfies certain restrictions.

There are several ways in which we can restrict the format of Context Free Grammar (CFG) without reducing the generation power of Context Free Grammar (CFG).

Before simplifying the grammar, let us consider the following Grammar $G = (\{S, A, B, E\}, \{a, b, c\}, P, S)$, where,

P is given by

$$S \rightarrow AB$$
$$A \rightarrow a$$
$$B \rightarrow b$$
$$B \rightarrow C$$
$$E \rightarrow C \mid \wedge$$

Looking at the above example, it is clear that

1. There are some non-terminal symbols which do not derive any terminal string, for example, C does not derive any string.
2. Some symbols (Terminals and Non–terminals) do not appear in any sentential form. For example, E and C can not appear in any sentential form, as, these are not reachable from the start symbol S.
3. The production $E \rightarrow \wedge$ is a null production and
4. The production $B \rightarrow C$ simply replaces B by C and is a unit production.

Next section deals with simplifying the context free grammar using above mentioned criteria.

4.5.2 Construction of Reduced Grammar (Eliminating Useless Symbols)

The reduction of the CFG involves following steps :

1. Identify **non-generating symbols** in the given Context Free Grammar (CFG) and eliminate those production which contain non-generating symbols.
2. Identify non-reachable symbols in the grammar and eliminate those productions which contain non-reachable symbols.

The grammar which we get after eliminating the productions corresponding to the non-generating and non-reachable symbols, is a useful Context Free Grammar (CFG).

Example 4.8 : Consider the Context Free Grammar (CFG) G

$$S \rightarrow AB \mid a$$

$$A \rightarrow b$$

Eliminate the useless symbols from the above Context Free Grammar (CFG).

Solution :

Given :

$$S \rightarrow AB \mid a$$

$$A \rightarrow b$$

1. Identify non-generating symbols

Observing each production in the CFG, it becomes very clear that B does not derive terminal string while A and S both derive the terminal string, i.e., $A \rightarrow b$ and $S \rightarrow a$ respectively. Hence, B is non-generating.

$\therefore$ We remove the production $S \rightarrow AB$ from the grammar, now the CFG becomes

$$S \rightarrow a$$

$$A \rightarrow b$$

2. Identify non-reachable symbols

Here, A is non-reachable symbol, as it can not be reached by starting symbols S. Hence, we remove the production.

$$A \rightarrow b$$

Now, the CFG becomes

$$S \rightarrow a$$

Which is the required Reduced Grammar.

Example 4.9 : Consider the following Context Free Grammar (CFG) G whose productions are :

$$S \rightarrow AB \mid CA$$

$$A \rightarrow a$$

$$B \rightarrow BC \mid AB$$

$$C \rightarrow aB \mid b$$

Find the reduced grammar equivalent to the above grammar G.

Solution : Given

$$S \rightarrow AB \mid CA$$

$$A \rightarrow a$$

$$B \rightarrow BC \mid AB$$

$$C \rightarrow aB/b$$

1. Identify non-generating symbols

(i) Non-terminal A **is generating** as there is a production $A \rightarrow a$.

(ii) Non-terminal C **is generating** as there is a production $C \rightarrow b$.

(iii) Non-terminal S **is generating** as there is a production. $S \rightarrow CA$ and non-terminals C and A both derive the terminal string, i.e., $C \rightarrow$ b and $A \rightarrow$ a.

(iv) Non-terminal B **is non-generating** because B does not derive any terminal string.

The set of non-generating symbol = $\{B\}$.

Removing the productions involving the non-generating symbol B, we get the following grammar.

$$S \rightarrow CA$$

$$A \rightarrow a$$

$$C \rightarrow b$$

2. Identify non-reachable symbols

(i) Non-terminal S is reachable as it is the **start symbol.**

(ii) Non-terminal A and C are also reachable as there is a production

$$S \rightarrow CA.$$

Therefore, no production is removed.

$\therefore$ The required reduced grammar is

$$S \rightarrow CA$$

$$A \rightarrow a$$

$$C \rightarrow b$$

Example 4.10 : Consider the context free grammar G whose productions are given as:

$$S \rightarrow XY \mid 0$$

$$X \rightarrow 1$$

Find the reduced grammar equivalent to the above grammar G.

Solution :

Given :

$$S \rightarrow XY \mid 0$$

$$X \rightarrow 1$$

1. Identify non-generating symbols :

(i) Non-terminal X **is generating** as there is a production $X \rightarrow 1$.

(ii) Non-terminal S **is also generating** as there is a production $S \rightarrow 0$.

(iii) Non-terminal **Y is non-generating** as Y does not derive any terminal string

$\therefore$ Set of non-generating symbols = $\{Y\}$.

Removing the productions involving the non-generating symbol Y, we get,

$$S \rightarrow 0$$

$$X \rightarrow 1$$

2. Identify non-reachable symbols :

(i) S is included in the reachable symbols as S is the start symbol.

(ii) X is non-reachable as there is no way to reach X from S.

Set of non-reachable symbols = $\{X\}$.

So, the production corrosponding to X, i.e., $X \rightarrow 1$ is removed from the final grammar.

$\therefore$ The reduced grammar is :

$$S \rightarrow 0$$

Example 4.11 : Remove useless symbols from the following context free grammar G whose productions are:

$$S \rightarrow aA \mid bB$$

$$A \rightarrow aA \mid a$$

$$B \rightarrow bB$$

$$D \rightarrow ab \mid Ea$$

$$E \rightarrow aC \mid d$$

Solution : Given :

$$S \rightarrow aA \mid bB$$

$$A \rightarrow aA \mid a$$

$$B \rightarrow bB$$

$$D \rightarrow ab \mid Ea$$

$$E \rightarrow aC \mid d$$

1. Identify non-generating symbols.

(i) Non-terminal A **is generating** as there is a production $A \rightarrow a$.

(ii) Non-terminal D **is generating** as there is a production $D \rightarrow ab$

(iii) Non-terminal E **is generating** as there is a production $E \rightarrow d$.

(iv) Non-terminal S **is also generating** as there is a production $S \rightarrow aA$ and non terminal A on RHS is a generating symbol.

(v) Non-terminal B **is non-generating** as the production $B \rightarrow bB$ will be continuously in loop and will not derive any non terminal.

(vi) Non-terminal C **is non-generating** as it does not derive any terminal string.

$\therefore$ Set of non-generating symbols = $\{B, C\}$.

Removing the productions involving the non-generating symbol B and C, we get,

$$S \rightarrow aA$$
$$A \rightarrow aA \mid a$$
$$D \rightarrow ab \mid Ea$$
$$E \rightarrow d$$

2. Identify non-reachable symbols :

(i) S is included in reachable symbol **as S is the start symbol.**

(ii) $S \rightarrow aA$ is a production means **S derives non-terminal** so A is also reachable.

(iii) A does not derive any non-terminal symbol

$\therefore$ Set of **reachable symbol** = $\{S, A\}$ and

Set of **non-reachable symbol** = $\{D, E\}$.

On removing the productions corrosponding to the **non-reachable symbols D and E**, we get

$$S \rightarrow aA$$
$$A \rightarrow aA \mid a$$

Which is the reduced grammar.

Example 4.12 : Consider the context-free grammar G whose productions are given as:

$$S \rightarrow aS \mid A \mid C$$
$$A \rightarrow a$$
$$B \rightarrow aa$$
$$C \rightarrow aCb$$

Find the reduced grammar equivalent to the grammar G.

Solution : Given :

$$S \rightarrow aS \mid A \mid C$$
$$A \rightarrow a$$
$$B \rightarrow aa$$
$$C \rightarrow aCb$$

1. Identify the non-generating symbol :

(i) Non-terminal A **is generating** as there is a production $A \rightarrow a$.

(ii) Non-terminal B **is generating** as there is a production $B \rightarrow aa$.

(iii) Non-terminal S **is generating** as there is a production $S \rightarrow A$ and non-terminal A on RHS is generating.

(iv) Non-terminal C **is non-generating** as the production $C \rightarrow aCb$ will be continuously in the loop and does not derive any string.

$\therefore$ **Set of non-generating symbol =** $\{C\}$.

Removing the production involving the non-generating symbol C, we get

$$S \rightarrow aS \mid A$$
$$A \rightarrow a$$
$$B \rightarrow aa$$

2. Identify non reachable symbols :

(i) S is included in reachable symbol as it is the **start symbol.**

(ii) $S \rightarrow A$ is a production, i.e., S derives non-terminal A so A **is also reachable.**

(iii) A does not derive any non-terminal symbol.

$\therefore$ **Set of reachable symbol** = $\{S, A\}$, and

Set of non-reachable symbol = $\{B\}$.

On removing the productions corrosponding to the **non-reachable symbols** B, we get

$$S \rightarrow aS \mid A$$
$$A \rightarrow a$$

Which is the required reduced grammar.

Example 4.13 : Eliminate the useless symbols from the following grammar.

$$S \rightarrow aA \mid a \mid Bb \mid cC$$
$$A \rightarrow aB$$
$$B \rightarrow a \mid Aa$$
$$C \rightarrow cCD$$
$$D \rightarrow ddd$$

Solution : Given :

$$S \rightarrow aA \mid a \mid Bb \mid cC$$
$$A \rightarrow aB$$
$$B \rightarrow a \mid Aa$$
$$C \rightarrow cCD$$
$$D \rightarrow ddd$$

1. Identify non-generating symbols

(i) Non-terminal D **is a generating** symbol as there is a production $D \rightarrow ddd$.

(ii) Non-terminal B **is a generating** symbol as there is a production $B \rightarrow a$.

(iii) Non-terminal A **is generating** as there is a production $A \rightarrow aB$ and non-terminal B on RHS is generating symbol.

(iv) Non-terminal S **is generating** as there is a production $S \to a$.

(v) Non-terminal C **is non-generating** as the production $C \to cCD$ will be in loop and does not derive string.

(C on RHS can not be replaced by any terminal).

$\therefore$ Set of non-generating symbol = $\{C\}$.

Removing the production involving the non-generating symbol C, we get,

$$S \to aA \mid a \mid Bb$$

$$A \to aB$$

$$B \to a \mid Aa$$

$$D \to ddd$$

2. Identify non-reachable symbol :

(i) S is included in the reachable symbol as **S is the start symbol.**

(ii) $S \to aA$ is a production, i.e., S derives non-terminal A, hence A is also **reachable.**

(iii) $S \to Bb$ is a production so, B is also **reachable.**

(iv) Non-terminal S, A and B does not derive any new non-terminal.

$\therefore$ **Set of reachable symbol** = $\{S, A, B\}$ and

Set of non-reachable symbol = $\{D\}$.

On reamoving the productions corrosponding to the non-reachable symbol D, we get,

$$S \to aA \mid a \mid Bb$$

$$A \to aB$$

$$B \to a \mid Aa$$

Which is the required reduced grammar.

4.5.3 Elimination of Null Productions

A production of the form $A \to \wedge$, where A is a non-terminal symbol or variable is called a null-production.

Nullable Variable

A variable in context free grammar is called nullable, if $A \xrightarrow{*} \wedge$ i.e., we can derive $\wedge$ in **zero** or **more** steps.

Procedure to eliminate null-productions

1. Find the productions of the form $A \xrightarrow{*} \wedge$ for each $A \in V_N$, i.e., find all nullable variables.
2. For each A obtained in step 1, find the productions in the set of production rules

whose right hand side contains A and replace each A by $\wedge$ symbol. The new productions are added in resultant grammar. The production obtained in this way are included in the grammar which is free from $\wedge$-production.

3. Any production in the given Context Free Grammar whose RHS does not contain any nullable variable is also included in new grammar.

Example 4.14 : Consider the following Context Free Grammar (CFG) whose productions are:

$$S \rightarrow aA$$
$$A \rightarrow b / \wedge$$

Eliminate the null productions.

Solution : Given :

$$S \rightarrow aA$$
$$A \rightarrow b / \wedge$$

1. Nullable variables

Since, $A \rightarrow \wedge$ is a production, therefore, A is the nullable variable.

$\therefore$ The set of nullable variable = $\{A\}$.

2. The productions whose RHS is a nullable variable (Here, it is A) is $S \rightarrow aA$

Replace A by $\wedge$, we get,

$$S \rightarrow a \qquad \text{....(1)}$$
$$S \rightarrow aA \qquad \text{(original production)} \qquad \text{....(2)}$$

3. The productions whose RHS **does not** contain any **nullable variable :**

$$A \rightarrow b \qquad \text{....(3)}$$

$\therefore$ The resultant grammar which has no $\wedge$ productions :

$$S \rightarrow a\,A$$
$$S \rightarrow a$$
$$A \rightarrow b$$

or

$$S \rightarrow aA / a$$
$$A \rightarrow b$$

Example 4.15 : Consider the Context Free Grammar G whose productions are:

$$S \rightarrow aS / AB$$
$$A \rightarrow \wedge$$
$$B \rightarrow \wedge$$
$$D \rightarrow b$$

Construct the equivalent grammar without null productions.

Solution : Given

$$S \to aS / AB$$
$$A \to \wedge$$
$$B \to \wedge$$
$$D \to b.$$

1. Nullable Variables

(i) $A \to \wedge$ is a production, therefore, ***A* is nullable** variable

(ii) $B \to \wedge$ is a production, therefore, ***B* is a nullable** variable

(iii) $S \to AB$ is a production and A and B are both nullable variable, therefore, ***S* is also nullable variable**

$\therefore$ Set of nullable variable = $\{S, A, B\}$.

2. The productions whose RHS is a nullable variable :

$$S \to aS$$
$$S \to AB$$

(i) For $S \to a\ S$ replace S by $\wedge$ in RHS, We get,

$$S \to a \quad \text{....(1)}$$
$$S \to aS \quad \text{(original production)} \quad \text{....(2)}$$

(ii) For $S \to AB$ in this case, we **can not** erase both nullable variables A and B in $S \to AB$ as we will get $S \to \wedge$. Hence, we get two productions.

$$S \to A \quad \text{....(3)}$$
$$S \to B \quad \text{....(4)}$$

3. The productions whose RHS does not contain nullable variables :

$$D \to b \quad \text{....(5)}$$

$\therefore$ The resultant grammar which has no-null productions is the grammar having productions from equation (1) to (5), i.e.,

$$S \to aS$$
$$S \to a$$
$$S \to A$$
$$S \to B$$
$$D \to b$$

or

$$S \to aS / a / A / B$$
$$D \to b$$

Example 4.16 : Consider the CFG whose productions are given as :

$$S \to XYX$$

$$X \to 0X/\wedge$$
$$Y \to 1Y/\wedge$$

Eliminate the null productions from above grammar.

Solution : Given :

$$S \to XYX$$
$$X \to OX/\wedge$$
$$Y \to 1Y/\wedge$$

1. Find nullable variables

$\because$ $X \to \wedge$ and

$Y \to \wedge$

$\therefore$ set of nullable variable = $\{X, Y\}$

2. The productions whose RHS is nullable variable :

(i) $S \to XYX$

In this case, we can not erase both nullable variable as we will get $S \to \wedge$. We can erase at most two variables, hence, we get following set of productions.

$S \to XY$ ....(1)

$S \to YX$ ....(2)

$S \to XX$ ....(3)

$S \to X$ ....(4)

$S \to Y$ ....(5)

(ii) $X \to 0X$

Replacing X on RHS by $\wedge$, we get

$X \to 0$ ....(6)

$X \to 0X$ (Original Production) ....(7)

(iii) $Y \to 1Y$

Replacing Y on RHS by $\wedge$, we get,

$Y \to 1$ ....(8)

$Y \to 1Y$ (Original Production) ....(9)

3. The production whose RHS does not contain any nullable variable :

No production exist under this category.

The **resultant grammar** which is free from null production is the set of production from (i) to (ix), i.e.,

$$S \to XY/YX/XX/X/Y$$
$$X \to 0X/0$$
$$Y \to 1Y/1$$

Example 4.17 : Eliminate the ∧-productions from the CFG given below.

$$A \rightarrow 0B1/1B1$$
$$B \rightarrow 0B/1B/\wedge$$

Solution : Given

$$A \rightarrow 0B1/1B1$$
$$B \rightarrow 0B/1B/\wedge$$

1. Find nullable variable :

Since, $B \rightarrow \wedge$ is a production, therefore, B is a nullable variable

The set of nullable variable = $\{B\}$

2. The productions whose RHS is a nullable variable :

(i) $A \rightarrow 0B1$

Replacing b on RHS by ∧, we get

$$A \rightarrow 01 \quad ....(1)$$
$$A \rightarrow 0B1 \quad \text{(Original Production)} \quad ....(2)$$

(ii) $A \rightarrow 1B1$

Replacing B on RHS by ∧, we get,

$$A \rightarrow 11 \quad ....(3)$$
$$A \rightarrow 1B1 \quad \text{(Original Production)} \quad ....(4)$$

(iv) $B \rightarrow 0B$

Replacing B on RHS by ∧, we get

$$B \rightarrow 0 \quad ....(5)$$
$$B \rightarrow 0B \quad \text{(Original Production)} \quad ....(6)$$

(iv) $B \rightarrow 1B$

Replacing B on RHS by ∧, we get

$$B \rightarrow 1 \quad ....(7)$$
$$B \rightarrow 1B \quad \text{(Original Production)} \quad ....(8)$$

(3) Productions whose RHS does not contain any nullable variable :

No production exist under this category.

The resultant grammar which is free from null move contains the set of productions from equation (1) to (8), i.e.,

$$A \rightarrow 01/0B1/11/1B1$$
$$B \rightarrow 0/0B/1/1B$$

Example 4.18 : Eliminate the ∧-productions from the grammar whose production rules are given below :

$$S \rightarrow aSa$$
$$S \rightarrow bSb$$
$$S \rightarrow \wedge$$

Solution : Given :

$$S \rightarrow aSa$$
$$S \rightarrow bSb$$
$$S \rightarrow \wedge$$

1. Find nullable variable :

$S \rightarrow \wedge$ is a production so S is a nullable variable.

$\therefore$ set of nullable variable = $\{S\}$

(2) Productions whose RHS contain the nullable variable :

(i) $S \rightarrow aSa$

Replacing S on RHS by $\wedge$, we get

$$S \rightarrow aa \quad ...(1)$$

$S \rightarrow aSa$ (Original Production) ....(2)

(ii) $S \rightarrow bSb$

Replacing S on RHS by $\wedge$, we get,

$$S \rightarrow bb \quad ....(3)$$

$S \rightarrow bSb$ (Original Production) ....(4)

3. There is **no production** in the given grammar whose **RHS does not contain any nullable** variable.

$\therefore$ The resultant grammar which is free from null move contains the set of productions from equation (1) to (4) i.e.,

$$S \rightarrow aa/aSa/bb/bSb$$

Example 4.19 : For the CFG given below, remove the null productions.

$$S \rightarrow a \mid Ab \mid aBa$$
$$A \rightarrow b \mid \wedge$$
$$B \rightarrow b \mid A$$

Solution : Given :

$$S \rightarrow a \mid Ab \mid aBa$$
$$A \rightarrow b/\wedge$$
$$B \rightarrow b/A$$

1. Find nullable variables :

$A \to \wedge$ is a production. So A is a nullable variable.

As A is nullable variable and there is a production $B \to A$ so B is also a nullable variable

$\therefore$ set of nullable variable = $\{A, B\}$

(2) Production whose RHS contains the nullable variable

(i) $S \to Ab$

Replacing A by $\wedge$ on RHS, we get,

$$S \to b \quad \text{....(1)}$$

$$S \to Ab \quad \text{(Original Production)} \quad \text{....(2)}$$

(ii) $S \to aBa$

Replacing B by $\wedge$ on RHS, we get,

$$S \to aa \quad \text{....(3)}$$

$$S \to aBa \quad \text{(Original Production)} \quad \text{....(4)}$$

(iii) $B \to A$

This production is deleted from the final set of production rule as it can not produce any useful language of the grammar.

(3) Production whose RHS does not contain any nullable variable :

$$S \to a \quad \text{....(5)}$$

$$A \to b \quad \text{....(6)}$$

$$B \to b \quad \text{....(7)}$$

The resultant grammar which is free from null move contains the set of production from equation (1) to (7) i.e.,

$$S \to b \mid Ab \mid aa \mid aBa \mid a$$

$$A \to b$$

$$B \to b$$

Example 4.20 : Consider the CFG given below :

$$S \to XY$$

$$S \to Zb$$

$$Y \to BW$$

$$Z \to AB$$

$$W \to Z$$

$$A \to aA/bA/\wedge$$

$$B \to Ba/Bb/\wedge$$

Elimiate the null productions from above grammar.

Solution : Given :

$$S \to XY$$
$$S \to Zb$$
$$Y \to BW$$
$$Z \to AB$$
$$W \to Z$$
$$A \to aA/bA/\wedge$$
$$B \to Ba/Bb/\wedge$$

1. Find nullable variable :

(i) $A \to \wedge$ and $B \to \wedge$ are the productions so A **and** B **are nullable variables.**

(ii) As A and B are nullable variable and there is a production $Z \to AB$, so Z is also a nullable variable.

(iii) There is a production $W \to Z$ and Z is a nullable variable so W is also a nullable variable.

$\therefore$ Set of nullable variable = $\{A, B, Z, W\}$

(2) Productions whose RHS contain the nullable variable :

(i) $X \to Zb$, Z is replaced by $\wedge$ on RHS, we get

$$X \to b \qquad \text{....(1)}$$
$$X \to Zb \qquad \text{(Original Production)} \qquad \text{....(2)}$$

(ii) $Y \to BW$ as both B and W on RHS are nullable varible, we can not erase both the variables, Hence, we get **two** productions,

$$Y \to B \qquad \text{....(3)}$$
$$Y \to W \qquad \text{....(4)}$$

(iii) $Z \to AB$ again A and B both are nullable variables, we can not erase both the variables, therefore, it gives

$$Z \to A \qquad \text{....(5)}$$
$$Z \to B \qquad \text{....(6)}$$

(iv) $W \to Z$ this production is deleted from the set of productions as it can not produce any useful language of the grammar.

(v) $A \to aA$ replacing A on RHS by $\wedge$, gives

$$A \to a \qquad \text{....(7)}$$
$$A \to aA \qquad \text{(Original Production)} \qquad \text{....(8)}$$

(vi) $A \to bA$ replacing A on RHS by $\wedge$, gives

$$A \to b \qquad \text{....(9)}$$
$$A \to bA \qquad \text{(Original Production)} \qquad \text{....(10)}$$

(vii) $B \rightarrow Ba$ replacing b by $\wedge$ on RHS, we get

$$B \rightarrow a \quad \text{....(11)}$$

$$B \rightarrow Ba \quad \text{(Original Production)} \quad \text{....(12)}$$

(viii) $B \rightarrow Bb$ replacing B on RHS by $\wedge$, gives,

$$B \rightarrow b \quad \text{....(13)}$$

$$B \rightarrow Bb \quad \text{(Original Production)} \quad \text{....(14)}$$

(3) Productions whose RHS does not contain any nullable variable :

$$S \rightarrow XY \quad \text{....(15)}$$

The resultant grammar which is free from null move contains the set of productions from equation (1) to (15). i.e.,

$$X \rightarrow b/Zb$$
$$Y \rightarrow B/W$$
$$Z \rightarrow A/B$$
$$A \rightarrow a/aA/b/bA$$
$$B \rightarrow a/Ba/b/Bb$$
$$S \rightarrow XY$$

Example 4.21 : Eliminate the $\wedge$-productions from the CFG given below :

$$\boldsymbol{S \rightarrow P0Q/QQ/0R/RQP}$$
$$\boldsymbol{P \rightarrow R0/1R/RR/RQP}$$
$$\boldsymbol{Q \rightarrow Q0/PQ/\wedge}$$
$$\boldsymbol{R \rightarrow 0P/QQQ}$$

Solution : Given

$$S \rightarrow P0Q/QQ/0R/RQP$$
$$P \rightarrow R0/1R/RR/RQP$$
$$Q \rightarrow Q0/PQ/\wedge$$
$$R \rightarrow 0P/QQQ$$

1. Find nullable variables :

(i) $Q \rightarrow \wedge$ is a production so $\boldsymbol{Q}$ is nullable variable.

(ii) $R \rightarrow QQQ$ is a production and Q is a nullable variable so $\boldsymbol{R}$ is also a nullable variable.

(iii) As R is nullable variable and there is a production $P \rightarrow RR$, therefore, P is also nullable variable

(iv) As Q is a nullable variable and there is a production $S \rightarrow QQ$, therefore, S is also nullable variable.

The set of nullable variable = $\{\boldsymbol{S, P, Q, R}\}$

(2) Productions whose RHS contains the nullable variable :

(i) $S \rightarrow P0Q$ gives

$S \rightarrow 0Q$ ....(1)

$S \rightarrow P0$ ....(2)

$S \rightarrow 0$ ....(3)

$S \rightarrow P0Q$ (Original production) ....(4)

(ii) $S \rightarrow QQ$ gives

$S \rightarrow Q$ ....(5)

(Since, it can not erase both variable or RHS).

(iii) $S \rightarrow 0R$ gives

$S \rightarrow 0$ ....(6)

$S \rightarrow 0R$ (Original Production) ....(7)

(iv) $S \rightarrow RQP$ gives

$S \rightarrow RQ$ ....(8)

$S \rightarrow QP$ ....(9)

$S \rightarrow RP$ ....(10)

$S \rightarrow R$ ....(11)

$S \rightarrow Q$ ....(12)

$S \rightarrow P$ ....(13)

(v) $P \rightarrow R0$ gives

$P \rightarrow 0$ ....(14)

$P \rightarrow R0$ (Original Production) ....(15)

(vi) $P \rightarrow 1R$ gives

$P \rightarrow 1$ ....(16)

$P \rightarrow 1R$ (Original Production) ....(17)

(vii) $P \rightarrow RR$ gives

$P \rightarrow R$ ....(18)

(Since, it can not erase both variable on RHS).

(viii) $P \rightarrow RQP$ gives

$P \rightarrow RQ$ ....(19)

$P \rightarrow QP$ ...(20)

$P \rightarrow RP$ ....(21)

$P \rightarrow R$ ....(22)

$P \rightarrow Q$ ....(23)

(ix) $Q \rightarrow Q0$ gives

$$Q \rightarrow 0 \quad \text{....(24)}$$

$$Q \rightarrow Q0 \quad \text{(Original Production)} \quad \text{....(25)}$$

(x) $Q \rightarrow PQ$ gives

$$Q \rightarrow P \quad \text{....(26)}$$

(xi) $R \rightarrow 0P$ gives

$$R \rightarrow 0 \quad \text{....(27)}$$

$$R \rightarrow 0P \quad \text{(Original Production)} \quad \text{....(28)}$$

(xii) $R \rightarrow QQQ$ gives

$$R \rightarrow QQ \quad \text{....(29)}$$

$$R \rightarrow Q \quad \text{....(30)}$$

Production from (1) to (30) form the required grammar which is free from null productions.

4.5.4 Elimination of Unit Productions

Definition

A production in this context free grammar G of the form $A \rightarrow B$ where A and B are non-terminals/variables in grammer G.

Procedure to eliminate Unit Production :

1. **Construction of Unit Pairs :** If there is a production $A \rightarrow B$ and $B \rightarrow C$ in the given Context Free Grammar (CFG) than make the unit pair $A \rightarrow C$ and apply this **chain rule** until we get non-unit production $X \rightarrow \alpha$.
2. Replace these unit productions by a single production $A \rightarrow \alpha$. Repeat above steps for each non-terminal in the given CFG.
3. Add all non-unit productions in the resultant grammar.

Example 4.22 : Consider the following Context Free Grammar (CFG) G whose productions are given below :

$$S \rightarrow A\ B$$
$$A \rightarrow a$$
$$B \rightarrow C \mid b$$
$$C \rightarrow D$$
$$D \rightarrow E$$
$$E \rightarrow a$$

Eliminate unit productions and find the resultant grammar.

Solution : Given

$$S \rightarrow A\ B$$
$$A \rightarrow a$$

$$B \to C / b$$
$$C \to D$$
$$D \to E$$
$$E \to a$$

1. Construction of Unit pairs

(i) For non-terminals S and A there is no unit productions.

(ii) For unit production $B \to C$, we have

$$B \to C$$
$$C \to D$$
$$D \to E$$
$$E \to a$$

Replace these unit production by single production, we get,

$$B \to a \quad \text{....(1)}$$

(iii) For unit production $\boldsymbol{C \to D}$, we have

$$C \to D$$
$$D \to E$$
$$E \to a$$

Replace these unit productions by single production, we get,

$$\boldsymbol{C \to a} \quad \text{....(2)}$$

(iv) For unit production $D \to E$, we have

$$D \to E$$
$$E \to a$$

Replace these unit production by the single production, we get,

$$D \to a \quad \text{....(3)}$$

(v) For E there is no unit production.

2. Non-unit productions in the given Context Free Grammar (CFG) :

$$S \to A\ B \quad \text{....(4)}$$
$$A \to a \quad \text{....(5)}$$
$$B \to b \quad \text{....(6)}$$
$$E \to a \quad \text{....(7)}$$

Therefore, the set of production rules in the resultant grammar which is free from unit productions consists of set of rules as in equation from (1) to (7), i.e.,

$$S \to AB$$
$$A \to a$$
$$B \to b$$

$$E \to a$$
$$B \to a$$
$$C \to a$$
$$D \to a$$

or

$$S \to A\ B$$
$$A \to a$$
$$B \to a \mid b$$
$$C \to a$$
$$D \to a$$
$$E \to a$$

Example 4.23 : Eliminate the unit productions from the Context-free grammar G given below :

$$\boldsymbol{S \to 0A \mid 1B \mid C}$$
$$\boldsymbol{A \to 0S \mid 00}$$
$$\boldsymbol{B \to 1 \mid A}$$
$$\boldsymbol{C \to 01}$$

Solution : Given :

$$S \to 0A \mid 1B \mid C$$
$$A \to 0S \mid 00$$
$$B \to 1 \mid A$$
$$C \to 01$$

1. Construction of unit pairs :

(i) For the **unit production $S \to C$**, we have

$$S \to C$$
$$C \to 01$$

Replacing this pair of unit production by single production, we get,

$$S \to 01 \quad \text{....(1)}$$

(ii) For the **unit production $B \to A$**, we have

$$B \to A$$
$$A \to 0S \mid 00$$

Replacing above pairs by single production, we get,

$$B \to 0S \mid 00 \quad \text{....(2)}$$

There is **no other unit production** in the grammar.

2. Non-unit productions in the CFG :

$$S \rightarrow 0A / 1B \quad \text{....(3)}$$

$$A \rightarrow 0S / 00 \quad \text{....(4)}$$

$$B \rightarrow 1 \quad \text{....(5)}$$

$$C \rightarrow 01 \quad \text{....(6)}$$

Combining the productions from (1) to (6), we get the required set of productions which is free from unit production.

$$S \rightarrow 0A / 1B / 01$$

$$A \rightarrow 0S / 00$$

$$B \rightarrow 1 / 0S / 00$$

$$C \rightarrow 01$$

Example 4.24 : Eliminate the unit productions from the context-free grammar given below :

$$\boldsymbol{S \rightarrow AB / A}$$

$$\boldsymbol{A \rightarrow C / d}$$

$$\boldsymbol{C \rightarrow b}$$

Solution : Given :

$$S \rightarrow AB / A$$

$$A \rightarrow C / d$$

$$C \rightarrow b$$

1. Construction of unit pairs :

(i) For the **unit production** $S \rightarrow A$, we have

$$S \rightarrow A$$

$$A \rightarrow C$$

$$C \rightarrow b$$

Replacing these pairs of production by single production, we get,

$$S \rightarrow b \quad \text{....(1)}$$

(ii) For the **unit production** $A \rightarrow C$, we have,

$$A \rightarrow C$$

$$C \rightarrow b$$

Replacing the above pair by single production, we get,

$$A \rightarrow b \quad \text{....(2)}$$

There is no other unit production in the given *CFG*.

2. Non-unit production in the *CFG* :

$$S \rightarrow AB \quad \text{....(3)}$$

$$A \rightarrow d \quad \text{....(4)}$$

$$C \rightarrow b \quad \text{....(5)}$$

Productions from (1) to (5) form the required set of productions free from unit productions. Which is shown below :

$$S \rightarrow b \mid AB$$

$$A \rightarrow b \mid d$$

$$C \rightarrow b$$

4.6 Normal Forms

In context free grammar, each production $A \rightarrow \beta$ where $\beta \in (V_N \cup \Sigma)^*$, i.e., right hand side of each production can be any string of non-terminals or terminals.

When the productions in Context Free Grammar (CFG) satisfy certain restrictions, then grammar G is said to be in a **normal form**. This section deals with **two** normal forms namely, **Chomsky Normal Form (CNF)** and **Greibach Normal Form (GNF).**

4.6.1 Chomsky Normal Form

Chomsky normal form is a kind of context-free grammar in which we have restriction on length of RHS of the production and the type of symbol on RHS of the production.

More formally, it can be defined as :

4.6.1.1 Definition

A context free grammar G is said to be in Chomsky Normal Form (CNF) if each production in the grammar is of the form :

$$A \rightarrow a \text{ or}$$

$$A \rightarrow B\ C$$

i.e., right hand side of each production can have **single terminal** or **two non-terminal** (variables) **symbols.** $S \rightarrow \wedge$ is allowed in grammar G if $\wedge$ is in the language generated by grammar G.

4.6.1.2 Procedure To Convert CFG To CNF

1. Eliminate **null productions** from the given CFG.
2. Eliminate **unit productions** from the given CFG.
3. All the productions in the grammar of the form $A \rightarrow a$ and $A \rightarrow BC$ are included in the resultant grammar of CNF.

4. Elimination of terminal symbols on RHS.

(i) Let us consider the productions of type $A \rightarrow x_1\ x_2\ x_3\ .....\ x_n$ with some terminals $a \in \Sigma$ on RHS. Replace each terminal $a \in \Sigma$ on RHS by new non-terminal symbol X and add a production $X \rightarrow a$ in the resultant grammar of CNF.

Note : If $\wedge$ is in the language generated by grammar G then start state S does not appear on RHS of any production.

(ii) After replacing each terminal by non-terminal, the productions of the form $A \rightarrow BC$ are included in (CNF).

5. Now, all the productions of the form $A \rightarrow A_1\ A_2\ ....\ A_n$ where $n \geq 3$, i.e., right hand side of the production contain more than three consecutive non-terminal symbols, we add new productions.

$$A \rightarrow A_1\ \ B_1$$
$$B_1 \rightarrow A_2\ \ B_2$$
$$\vdots \qquad \vdots \quad \vdots$$
$$B_{n-3} \rightarrow A_{n-2}\ B_{n-2}$$
$$B_{n-2} \rightarrow A_{n-1}\ A_n$$

Where $B_1, B_2, ... B_{n-2}$ are the **new non-terminals** and are added in such a way that the length of terminals on RHS is always **exactly two.**

The productions obtained in step (3), (4) and (5) are the required productions for CNF.

Example 4.25 : Convert the following context-free grammar into Chomsky normal form.

$$S \rightarrow aaaaS$$
$$S \rightarrow aaaa$$

Solution :

Given :

$$S \rightarrow aaaaS$$
$$S \rightarrow aaaa$$

Step 1 : The context-free grammar does not contain any null production, so, we can skip this step.

Step 2 : **Elimination of unit production :** The given context-free grammar also does not contain any unit production so, we can skip this step.

Step 3 : **Productions of the form $A \rightarrow a$ and $A \rightarrow BC$:** No production under this category exist.

Step 4 : Elimination of terminal symbol on RHS :

(a) Production $S \rightarrow aaaaS$ yields

$$S \rightarrow AAAAS \qquad ....(1)$$

(A new non-terminal A is added due to terminal a)

and $A \rightarrow a$ ....(2)

(b) Production $S \rightarrow aaaa$ yields

$S \rightarrow AAAA$ ....(3)

Step 5 : Restriction on number of variables on RHS.

(a) Production (2) is in required form of CNF.

(b) For production (1), $S \rightarrow AAAAS$, we get,

$S \rightarrow AR_1$ (where, $R_1 = AAAS$) ....(4)

$R_1 \rightarrow AR_2$ (where, $R_2 = AAS$) ....(5)

$R_2 \rightarrow AR_3$ (where, $R_3 = AS$) ....(6)

$R_3 \rightarrow AS$ ....(7)

(c) For production (2), $S \rightarrow AAAA$, we can replace by

$S \rightarrow R_4R_5$ (where $R_4 = AA$) ....(8)

$R_4 \rightarrow AA$ ....(9)

$R_5 \rightarrow AA$ ....(10)

Note : Production (9) and (10) can be replaced by single production, and one of the production can be removed.

The productions of equation (2), (4), (5), (6), (7), (8) and (10) constitute the required productions of chomsky normal form.

Example 4.26 : Convert the following context-free grammar into equivalent Chomsky normal form:

$S \rightarrow aAbB$

$A \rightarrow aA/a$

$B \rightarrow bB/b$

Solution :

Given :

$S \rightarrow aAbB$

$A \rightarrow aA/a$

$B \rightarrow bB/b$

Step 1 : The context-free grammar does not contain any null production, so, we can skip this step.

Step 2 : **Elimination of unit production :** The given context-free grammar also does not contain any unit production so, we can skip this step.

Step 3 : Productions of the form $A \rightarrow a$ and $A \rightarrow BC$:

$A \rightarrow a$ ....(1)

$B \rightarrow b$ ....(2)

Step 4 : Elimination of terminal symbol on RHS :

(a) Production $S \rightarrow aAbB$ yields

$$S \rightarrow R_1AR_2B \quad \text{....(3)}$$

(b) Production $A \rightarrow aA$ yields

$$A \rightarrow R_1A \quad \text{....(4)}$$

(c) Production $B \rightarrow bB$ yields

$$B \rightarrow R_2B \quad \text{....(5)}$$

(d) Productions corrosponding to the new non-terminals R_1 and R_2 :

$$R_1 \rightarrow a \quad \text{....(6)}$$

$$R_2 \rightarrow b \quad \text{....(7)}$$

Step 5 : (a) Productions obtained in equation (1), (2), (4), (5), (6) and (7) are in required form of CNF.

(b) For production obtained in equation (3), $S \rightarrow R_1AR_2B$, we can add new production rules as follows :

$$S \rightarrow R_1R_3 \text{ (where, } R_3 = AR_2B) \quad \text{....(8)}$$

$$R_3 \rightarrow AR_4 \text{ (where, } R_4 = R_2B) \quad \text{....(9)}$$

$$R_4 \rightarrow R_2B \quad \text{....(10)}$$

The productions obtained in equation (1), (2), (4), (5), (6), (7), (8), (9) and (10) are the required productions of chomsky normal form.

Example 4.27 : Convert the following context free grammar in Chomsky normal form

$$S \rightarrow aSa$$
$$S \rightarrow bSb$$
$$S \rightarrow a$$
$$S \rightarrow b$$

Solution :

Given :

$$S \rightarrow aSa$$
$$S \rightarrow bSb$$
$$S \rightarrow a$$
$$S \rightarrow b$$

Step 1 : The context-free grammar does not contain any null production, so, we can skip this step.

Step 2 : **Elimination of unit production :** The given context-free grammar also does not contain any unit production so, we can skip this step.

Step 3 : Productions of the form $A \to a$ and $A \to BC$:

$$S \to a \quad \text{....(1)}$$

$$S \to b \quad \text{....(2)}$$

Step 4 : Elimination of terminal symbols on RHS :

(a) Production $S \to aSa$ yield

$$S \to R_1SR_1 \quad \text{....(3)}$$

(b) Production $S \to bSb$ yield

$$S \to R_2SR_2 \quad \text{....(4)}$$

(c) Productions corrosponding to the **new non-terminals R_1 and R_2 :**

$$R_1 \to a \quad \text{....(5)}$$

$$R_2 \to b \quad \text{....(6)}$$

Step 5 :

(a) Productions in equation (1), (2), (5) and (6) are in required from of CNF.

(b) For production $S \to R_1SR_1$, new productions are added as follows :

$$S \to R_1R_4 \text{ (where } R_4 = SR_1) \quad \text{....(7)}$$

$$R_4 \to SR_1 \quad \text{....(8)}$$

(c) For production $S \to R_2SR_4$, new productions are added as follows :

$$S \to R_2R_5 \text{ (where } R_5 = SR_4) \quad \text{....(9)}$$

$$R_5 \to SR_4 \quad \text{....(10)}$$

The productions of equation (1), (2), (5), (6), (7), (8), (9) and (10) form the required productions of CNF.

Example 4.28 : Consider the grammar $G = (\{S, A\}, \{a, b\}, P, S)$ where P consists of

$$S \to aAS \mid a$$

$$A \to SbA \mid SS \mid ba$$

Convert it into equivalent Chomsky normal form.

Solution : Given : $S \to aAS \mid a$

$$A \to SbA \mid SS \mid ba$$

Step 1 : The context-free grammar does not contain any null production, so, we can skip this step.

Step 2 : **Elimination of unit production :** The given context-free grammar also does not contain any unit production so, we can skip this step.

Step 3 : Productions of the form $A \to a$ and $A \to BC$:

$$S \rightarrow a \quad \text{....(1)}$$

$$A \rightarrow SS \quad \text{....(2)}$$

Step 4 : Elimination of terminal symbol on RHS :

(a) Production $S \rightarrow aAS$ yields

$$S \rightarrow R_1AS \quad \text{....(3)}$$

(b) Production $A \rightarrow SbA$ yields

$$A \rightarrow SR_2A \quad \text{....(4)}$$

(c) Production $A \rightarrow ba$ yields

$$A \rightarrow R_2R_1 \quad \text{....(5)}$$

Productions corrosponding to the new non-terminals R_1 and R_2 :

$$R_1 \rightarrow a \quad \text{....(6)}$$

$$R_2 \rightarrow b \quad \text{....(7)}$$

Step 5 :

(a) Production in equation (1), (2), (5), (6) and (7) are in required form of CNF.

(b) Production of equation (3) $S \rightarrow R_1 AS$ in changed as follows :

$$S \rightarrow R_1R_3 \text{ (where } R_3 = AS) \quad \text{....(8)}$$

$$R_3 \rightarrow AS \quad \text{....(9)}$$

(c) Production of equation (4), $A \rightarrow SR_2 A$ is changed as follows :

$$A \rightarrow SR_4 \text{ (where } R_4 = R_2A) \quad \text{....(10)}$$

$$R_4 \rightarrow R_2A \quad \text{....(11)}$$

(1), (2), (5), (6), (7), (8), (9), (10) and (11) are the required productions of the form CNF.

Example 4.29 : Convert the following CFG into equivalent CNF.

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

Solution : Given :

$$S \rightarrow aB$$

$$S \to bA$$
$$A \to a$$
$$A \to aS$$
$$A \to bAA$$
$$B \to b$$
$$B \to aS$$
$$B \to aBB$$

Step 1 : The context-free grammar does not contain any null production, so, we can skip this step.

Step 2 : **Elimination of unit production :** The given context-free grammar also does not contain any unit production so, we can skip this step.

Step 3 : Productions of the from $A \to a$ and $A \to BC$:

$$A \to a$$
$$B \to b \quad \text{....(1)}$$

Step 4 : Elimination of terminal symbol on RHS : ....(2)

(a) Production $S \to aB$ yields

$$S \to R_1B$$

(b) Production $S \to bA$ yields ....(3)

$$S \to R_2A$$

(c) Production $A \to aS$ yields ....(4)

$$A \to R_1S$$

(d) Production $A \to bAA$ yields ....(5)

$$A \to R_2AA$$

(e) Production $B \to aS$ yields ....(6)

$$B \to R_1S$$

(f) Production $B \to aBB$ yields ....(7)

$$B \to R_1BB$$

Productions corrosponding to the new non-terminals R_1 and R_2 : ....(8)

$$R_1 \to a \quad \text{....(9)}$$
$$R_2 \to b \quad \text{....(10)}$$

Step 5 :

(a) Production in equation (1), (2), (3), (4), (5), (7), (9) and (10) are in required form of CNF.

(b) For production in equation (6), $A \rightarrow A_2AA$, new productions are added ab follows

$$A \rightarrow R_2R_3 \text{ (where, } R_3 = AA) \quad \text{....(11)}$$

$$R_3 \rightarrow AA \quad \text{....(12)}$$

(c) For productions in equation (8), $B \rightarrow R_1BB$, new productions are added as follows :

$$B \rightarrow R_1R_4 \text{ (where } R_4 = BB) \quad \text{....(13)}$$

$$R_4 \rightarrow BB \quad \text{....(14)}$$

(1), (2), (3), (4), (5), (7), (9), (10), (11), (12), (13) and (14) are the required productions of the form CNF.

Example 4.30 : Convert the following grammar into equivalent chomsky normal form:

$$S \rightarrow \sim S/[S \supset S] \ / \ p \ / \ q$$

Solution : Given : $S \rightarrow \sim S/[S \supset S] \ / \ p \ / \ q$

Step 1 : The context-free grammar does not contain any null production, so, we can skip this step.

Step 2 : **Elimination of unit production :** The given context-free grammar also does not contain any unit production so, we can skip this step.

Step 3 : Productions of the form $A \rightarrow a$ and $A \rightarrow BC$:

$$S \rightarrow p \quad \text{....(1)}$$

$$S \rightarrow q \quad \text{....(2)}$$

Step 4 : Elimination of terminal symbols on RHS :

(a) Production $S \rightarrow \sim S$ yields

$$S \rightarrow R_1S \quad \text{....(3)}$$

(b) Production $S \rightarrow [S \supset S]$ yields

$$S \rightarrow R_2SR_3SR_4 \quad \text{....(4)}$$

Productions corrosponding to the new non-terminals R_1, R_2, R_3 and R_4 :

$$R_1 \rightarrow \sim \quad \text{....(5)}$$

$$R_2 \rightarrow [\quad \text{....(6)}$$

$$R_3 \rightarrow \supset \quad \text{....(7)}$$

$$R_4 \rightarrow] \quad \text{....(8)}$$

Step 5 : (a) Productions in equation (1), (2), (3), (5), (6), (7) and (8) are in required form of CNF.

(b) For production obtained in equation (4), $S \rightarrow R_2SR_3SR_4$, new productions are added as follows :

$$S \rightarrow R_2R_5 \text{ (where } R_5 = SR_3SR_4) \quad \text{....(9)}$$

$$R_5 \rightarrow SR_6 \text{ (where } R_6 = R_3SR_4) \quad \ldots(10)$$

$$R_6 \rightarrow R_3R_7 \text{ (where } R_7 = SR_4) \quad \ldots(11)$$

$$R_7 \rightarrow SR_4 \quad \ldots(12)$$

(1), (2), (3), (5), (6), (7), (8), (9), (10), (11) and (12) are the required productions of CNF.

4.6.2 Greibach Normal Form (GNF)

GNF is another useful form in generating the set accepted by pushdown automaton.

4.6.2.1 Definition

A context free gramman G is said to be in GNF if every production is of the form $A \rightarrow a\alpha$ where, $\alpha \in V_N^*$ (i.e., α may be null), $a \in \Sigma$ and $S \rightarrow \wedge$ is allowed in grammar, if $\wedge$ is in the language generated by the grammar.

Before giving the procedure to convert the given CFG into equivalent GNF, we need to discuss two major problems, namely, left factoring and left recursion.

1. Left Factoring

Let $G = (V_N, \Sigma, P, S)$ be a Context Free Grammar. Let $A \rightarrow B\gamma$ be an A–production in P and $B \rightarrow \beta_1 / \beta_2 / \beta_3 / \ldots \beta_n$ be the B–productions. Then, it is always possible to replace B in $A \rightarrow B\gamma$ by β_i where, $1 \leq i \leq n$ in B-productions, i.e.,

$$A \rightarrow \beta_1\gamma / \beta_2\gamma / \ldots \beta_n\gamma$$

and

$$B \rightarrow \beta_1 / \beta_2 / \beta_3 / \ldots \beta_n$$

Note : This is useful for deleting a variable B as the first symbol on the RHS of some A-productions provided no B–productions has B as the first symbol on RHS.

2. Left Recursion

Let $G = (V_N, \Sigma, P, S)$ be a Context Free Grammar (CFG). The set of A-productions be

$A \rightarrow A\alpha_1 / A\alpha_2 / \ldots / A\alpha_m$ (i.e., A-productions starting with A) and

$A \rightarrow \beta_1 / \beta_2 / \beta_3 / \ldots / \beta_n$ (i.e., A-productions starting with symbol other than A)

Then, it is possible to remove immediate A from the above set of productions by introducing new variable Z as follows :

(i) The set of A–productions :

$$A \rightarrow \beta_1 / \beta_2 / \beta_3 / \ldots / \beta_n$$

$$A \rightarrow \beta_1 Z / \beta_2 Z / .... / \beta_n Z$$

(ii) The set of Z–productions

$$Z \rightarrow \alpha_1 / \alpha_2 / .... / \alpha_m$$

$$Z \rightarrow \alpha_1 Z / \alpha_2 Z / ..... / \alpha_m Z$$

Example 4.31 : Consider the Context Free Grammer (CFG) *G* whose productions are given below :

$$A \rightarrow aBD / bDB / c$$

$$A \rightarrow AB / AD$$

Remove left recursion from the grammar.

Solution : Given

$$A \rightarrow aBD / bDB / c$$

$$A \rightarrow AB \mid AD$$

Here, $\alpha_1 = B,\ \alpha_2 = D$

$\beta_1 = a\ B\ D,\ \beta_2 = b\ D\ B,\ \beta_3 = c$

From above definition

∴ New *A*–productions are :

$$A \rightarrow a\,B\,D / bDB / c$$

$$A \rightarrow aBDZ / bDBZ / c\,Z \text{ and}$$

New *Z*–productions are :

$$Z \rightarrow B / D$$

$$Z \rightarrow BZ / DZ$$

4.6.2.2 Procedure to Convert CFG into GNF

Step 1. Convert the given context-free grammar into equivalent Chomsky Normal Form (CNF).

Step 2. Rename all non-terminal symbols of the grammar as $A_1, A_2, ...A_n$ with $S = A_1$. Now, the grammar will become $G = ((A_1, A_2, ...A_n), \Sigma, P, A_1)$

Step 3. Now, seperate the productions

(i) Productions of the form $A_i \rightarrow a\gamma$ or $A_i \rightarrow A_j\gamma$ where $j > i$

(ii) Productions of the form $A_i \rightarrow A_j\gamma$ where $j \leq i$

Step 4. (i) If there is a production $A_1 \rightarrow A_1\gamma$, apply left recursion on this production. Similarly, apply left recursion on any predeution of the form $A_i \rightarrow A_j\gamma$ for $i = 1, 2, ...n$.

Thus, we get the productions of the form $A_i \rightarrow A_j\gamma$ where, $j > i$ and $i = 1, 2, \ldots n-1$.

Step 5. Apply left-recursion in A_n-Productions : This resulting A_n-Productions will be of form $A_n \rightarrow \alpha\gamma$ means these productions will be in required form.

Step 6. Modify A_i-productions for $i = 1, 2, \ldots, n-1$: Apply left recursion on production $A_{n-1}, A_{n-2}, \ldots A_2, A_1$.

Step 7. Modify Z_i-production : Apply left-factoring on the productions corrosponding to any newly introduced variable.

The productions obtained in step (5), (6) and (7) are the required production which are in GNF.

Example 4.32 : Consider the following CFG *G* whose productions are given below:

$$S \rightarrow AB$$
$$A \rightarrow BS/b$$
$$B \rightarrow SA/a$$

Convert it into equivalent GNF.

Solution : Given

$$S \rightarrow AB$$
$$A \rightarrow BS/b$$
$$B \rightarrow SA/a$$

1. The given CFG is in the CNF.

2. Rename Variables S, A and B as A_1, A_2 and A_3 respectively, we get

$$A_1 \rightarrow A_2A_3$$
$$A_2 \rightarrow A_3A_1/b$$
$$A_3 \rightarrow A_1A_2/a$$

(Here, number of variables, $n = 3$)

3. Seperation of Productions :

(i) Productions of the form $A_i \rightarrow \alpha\gamma$ or $A_i \rightarrow A_j\gamma$ where $j > i$ are in required form

$$A_1 \rightarrow A_2A_3$$
$$A_2 \rightarrow A_3A_1 \mid b$$
$$A_3 \rightarrow a$$

are in required form.

(ii) Productions of the form $A_i \rightarrow A_j\gamma$ where $j \leq i$, $A_3 \rightarrow A_1 A_2$

4. Apply left factoring on this production.

We get,

$$A_3 \rightarrow A_2 A_3 A_2 \qquad (\because A_1 \rightarrow A_2 A_3)$$

Apply once again left factoring on this production since $j \geq i$, we get,

$$A_3 \rightarrow A_3A_1A_2 / bA_1A_2 \qquad \left(\because A_3 \rightarrow A_1A_1 / b\right)$$

A_3-productions are :

$$A_3 \rightarrow a$$
$$A_3 \rightarrow bA_1A_2$$
$$A_3 \rightarrow A_3A_1A_2A_2$$

5. Apply left recursion to A_3-productions by introducing new variable Z_3, we get :

(i) A-productions :

$$A_3 \rightarrow a$$
$$A_3 \rightarrow bA_1A_2$$
$$A_3 \rightarrow aZ_3$$
$$A_3 \rightarrow bA_1A_2Z_3$$

(ii) Z-productions :

$$Z_3 \rightarrow A_1A_2A_2$$
$$Z_3 \rightarrow A_1A_2A_2Z_3$$

6. A_n-production :

(i) A_3-productions are :

$$A_3 \rightarrow bA_1A_2 / bA_1A_2Z_3 / a / aZ_3 \qquad \ldots(1)$$

(ii) A_2-productions are :

$$A_2 \rightarrow A_3A_2 / b$$

Among these productions, we retain $A_2 \rightarrow b$ and eliminate $A_2 \rightarrow A_3A_1$ by using left factoring, we get,

$$A_2 \rightarrow bA_1A_2A_1 / bA_1A_2Z_3A_1 / aA_1 / aZ_3A_1 \qquad \ldots(2)$$

(iii) A_1-productions are :

$$A_1 \rightarrow A_2A_3^*$$

Apply left factoring on above production, we get,

$$A_1 \rightarrow bA_1A_2A_1A_3 / bA_1A_2Z_3A_1A_3 / aA_1A_3 / aZ_3A_1A_3 \qquad \ldots(3)$$

7. Z-productions :

Modify the Z-production using left factoring, we get,

$$Z_3 \rightarrow bA_1A_2A_1A_3A_3A_2/bA_1A_2Z_3A_1A_3A_3A_2/aA_1A_3A_3A_2/ aZ_3A_1A_3A_3A_2$$

and $$Z_3 \rightarrow bA_1A_2A_1A_3A_3A_2Z_3/bA_1A_2Z_3A_1A_3A_3A_2Z_3/aA_1A_3A_3A_2Z_3/aZ_3A_1A_3A_3A_2Z_3 \qquad \ldots(4)$$

The prodcutions obtained in equation (1) to (4) constitute the productions of resultant grammar.

Example. 4.33 : Construct the grammar in Greibach normal form equivalent to the context free grammer given below

$$S \to AA \mid a,$$
$$A \to SS \mid b$$

Solution : Given :

$$S \to AA \mid a,$$
$$A \to SS \mid b$$

Step 1. The given grammar is in CNF so, skip this step.

Step 2. Rename variables S and A as A_1, and A_2, respectively, we get

$$A_1 \to A_2A_2 \mid a,$$
$$A_2 \to A_1A_1 \mid b$$

Here, the number of varaibles $(n) = 2$

Step 3. Now, seperate the productions :

(i) Productions of the form $A_i \to a\gamma$ or $A_i \to A_j\gamma$ where $j > i$

$$A_1 \to a$$
$$A_1 \to A_2A_2$$
$$A_2 \to b$$

(ii) Productions of the form $A_i \to A_j\gamma$ where $j \leq i$

$$A_2 \to A_1A_1$$

Step 4. Apply left factoring on A_n-production.

(i) $A_2 \to A_1A_1$ is replaced by

$$A_2 \to A_2A_2A_1 \qquad (\because \ A_1 \to A_2A_2)$$
$$A_2 \to aA_1 \qquad (\because \ A_1 \to a)$$

$\therefore$ A_n-productions are

$$A_2 \to A_2A_2A_1$$
$$A_2 \to aA_1$$

and

$$A_2 \to b$$

Step 5. Apply left recursion on A_n-productions : We introduce new variable Z_2 corrosponding to the variable A_2, we get,

(i) A_2-productions :

$$A_2 \to aA_1$$
$$A_2 \to b$$
$$A_2 \to aA_1Z_2$$

and $A_2 \rightarrow bZ_2$

(ii) Z_2-productions :

$$Z_2 \rightarrow A_2A_1$$
$$Z_2 \rightarrow A_2A_1Z_2$$

Step 6.(a) Modified A_n-productions :

$$A_2 \rightarrow aA_1$$
$$A_2 \rightarrow b$$
$$A_2 \rightarrow aA_1Z_2$$

and $A_2 \rightarrow bZ_2$

(b) A_{n-1}-productions :

$$A_1 \rightarrow a$$

and $A_1 \rightarrow A_2A_2$

Among A_1-productions, we retain $A_1 \rightarrow a$ which is in required form and eliminate $A_1 \rightarrow A_2A_2$ by using left factoring. The resultant A_1-productions are

$$A_1 \rightarrow aA_1A_2$$
$$A_1 \rightarrow bA_2$$
$$A_1 \rightarrow aA_1Z_2A_2$$
$$A_1 \rightarrow bZ_2A_2$$

Step 7. Modify Z-productions :

$$Z_2 \rightarrow aA_1A_1$$
$$Z_2 \rightarrow bA_1$$
$$Z_2 \rightarrow aA_1Z_2A_1$$
$$Z_2 \rightarrow bZ_2A_1$$
$$Z_2 \rightarrow aA_1A_1Z_2$$
$$Z_2 \rightarrow bA_1Z_2$$
$$Z_2 \rightarrow aA_1A_1Z_2$$
$$Z_2 \rightarrow bZ_2A_1Z_2$$

Hence, the equivalent grammar GNF is :

$$A_1 \rightarrow a \mid aA_1A_2 \mid bA_2 \mid aA_1Z_2A_2 \mid bZ_2A_2$$
$$A_2 \rightarrow aA_1 \mid b \mid aA_1Z_2 \mid bZ_2$$
$$Z_2 \rightarrow aA_1A_1 \mid bA_1 \mid aA_1Z_2A_1 \mid bZ_2A_1$$
$$Z_2 \rightarrow aA_1A_1Z_2 \mid bA_1Z_2 \mid aA_1Z_2A_1Z_2 \mid bZ_2A_1Z_2$$

Example 4.34 : Convert the following context-free grammar into equivalent GNF.

$$E \to E + T / T$$
$$T \to T * F / F$$
$$F \to (E) / a$$

Solution : Given :

$$E \to E + T / T$$
$$T \to T * F / F$$
$$F \to (E) / a$$

(1) Convert the grammar in CNF :

(A) First eliminate the unit productions :

(i) Construction of unit pairs

$\Rightarrow$ For production $E \to T$, we have

$$E \to T$$
$$T \to F$$
$$F \to a$$

Replacing this pair of unit productions by single production, we get,

$$E \to a \quad \ldots(1)$$

$\Rightarrow$ For production $T \to F$, we have

$$T \to F$$
$$F \to a \quad \ldots(2)$$

Replacing this pair of unit production by single production, we get, ...(3)

$$T \to a$$

There is no other unit pair in the grammar. ...(4)

(ii)Non-unit productions in CFG.

$$E \to E + T$$
$$T \to T * F \quad \ldots(5)$$
$$F \to (E) / a \quad \ldots(6)$$

The equivalent grammar without unit production is : ...(7)

$$E \to E + T / T * F / (E) / a$$
$$T \to T * F / (E) / a$$
$$F \to (E) / a$$

Note : $T \to F$ is replaced by $T \to (E)$ and $T \to a$. Similarly, $E \to T$ is replaced $E \to T * F$, $E \to (E)$ and $E \to a$.

(B) The grammar does not contain any null production.

(C) Productions of the form $A \rightarrow a$ and $A \rightarrow BC$

$$E \rightarrow a$$
$$T \rightarrow a$$
$$F \rightarrow a$$

(D) Elimination of terminal symbols in RHS

(i) Production $E \rightarrow E + T$ yields

$$E \rightarrow EAT$$

(ii) Production $E \rightarrow T * F$ yields

$$E \rightarrow TBF$$

(iii) Production $E \rightarrow (E)$ yields

$$E \rightarrow (EC$$

(iv) Productions corresponding to the new non-terminals A, B and C, we get,

$$A \rightarrow +$$
$$B \rightarrow *$$
$$C \rightarrow)$$

The modified productions are

$$E \rightarrow EAT \mid TBF \mid (EC \mid a$$
$$T \rightarrow TBF \mid (EC \mid a$$
$$A \rightarrow (EC \mid a$$
$$A \rightarrow +,\ B \rightarrow *,\ C \rightarrow)$$

(2) Rename variable A, B, C, F, T, E as A_1, A_2, A_3, A_4, A_5 and A_6 respectively, we get,

$$A_1 \rightarrow +$$
$$A_2 \rightarrow *$$
$$A_3 \rightarrow)$$
$$A_4 \rightarrow (A_6A_3 \mid a$$
$$A_5 \rightarrow A_5A_2A_4 \mid A_6A_3 \mid a$$
$$A_6 \rightarrow A_6A_1A_5 \mid A_5A_2A_4 \mid (A_6A_3 \mid a$$

(3) Seperate the productions :

(i) Productions of the form $A_i \rightarrow \alpha\gamma$ or $A_i \rightarrow A_j\gamma$ where, $j > i$,

$$A_1 \rightarrow +$$
$$A_2 \rightarrow *$$
$$A_3 \rightarrow)$$
$$A_4 \rightarrow (A_6A_3 \mid a$$
$$A_5 \rightarrow (A_6A_3 \mid a$$

$$A_6 \rightarrow (A_6A_3 \ / \ a \qquad \ldots(1)$$

(4) Apply left factoring on A_n-Production

(**Note :** Here, number of variables, $n = 6$)

$$A_6 \rightarrow A_6A_1A_5$$

(a)Apply left-recursion on A_5-Productions, $A_5 \rightarrow A_5A_2A_4$, we get,

$$A_5 \rightarrow (A_6A_3 \ / \ a$$

$$A_5 \rightarrow (A_6A_3Z_5 \ / \ aZ_5$$

and

$$Z_5 \rightarrow A_2A_4 \ / \ A_2A_4Z_5 \qquad \ldots(2)$$

(b) Apply left-factoring on A_6-productions, $A_6 \rightarrow A_5A_2A_4$, we get

$$A_6 \rightarrow (A_6A_3A_2A_4 \ / \ aA_2A_4 \ / \ (A_6A_3Z_5A_2A_4 \ / \ aZ_5A_2A_4$$

Now, A_n-productions are

$$A_6 \rightarrow (A_6A_3A_2A_4 \ / \ aA_2A_4 \ / \ (A_6A_3Z_5A_2A_4 \ / \ aZ_5A_2A_4$$

$$A_6 \rightarrow (A_6A_3 \ / \ a \qquad \ldots(3)$$

(5) Apply left recursion on A_n-Productions, $A_6 \rightarrow A_6A_1A_5$, we get,

$$A_6 \rightarrow (A_6A_3A_2A_4 \ / \ aA_2A_4 \ / \ A_6A_3Z_5A_2A_4$$

$$A_6 \rightarrow aZ_5A_2A_4 \ / \ (A_6A_3 \ / \ a$$

$$A_6 \rightarrow (A_6A_3A_2A_4A_6 \ / \ aA_2A_4Z_6 \ / \ (A_6A_3Z_5A_2A_4Z_6$$

$$A_6 \rightarrow aZ_6A_2A_4A_6 \ / \ (A_6A_3Z_6 \ / \ aZ_6 \qquad \ldots(4)$$

Z-productions are :'

$$Z_6 \rightarrow A_1A_5 \ / \ A_1A_5Z_6 \qquad \ldots(5)$$

(6) The productions A_{n-1}, A_{n-2}, ... A_1, i.e., A_5, A_4, A_3, A_2 and A_1 are in required form.

(7) Modified Z-Productions :

(i) Z_5-Productions :

$$Z_5 \rightarrow {}^*A_4 \ / \ {}^*A_4Z_5$$

(ii) Z_6-Productions : $\qquad \ldots(6)$

$$Z_6 \rightarrow {}^+A_5 \ / \ {}^+A_5Z_6$$

Productions of equation (1) to (7) constitute the resultant grammar. $\qquad \ldots(7)$

Example. 4.35 : Find the grammar in GNF equivalent to the context-free grammar G whose productions are given below :

$$A_1 \rightarrow A_2A_3$$

$$A_2 \rightarrow A_3A_1 \ / \ b$$

$$A_3 \rightarrow A_1A_2 \ / \ a$$

Solution : Given :

$$A_1 \rightarrow A_2A_3$$

$$A_2 \rightarrow A_3A_1 \ / \ b$$

$$A_3 \rightarrow A_1A_2 \ / \ a$$

Step 1. The given is already in CNF, we can skip this step.

Step 2. **Rename variables :** There is no need to rename the variables as these are already in sequence.

Step 3. Seperate the productions

(i) Productions of the form $A_i \to a\gamma$ or $A_i \to A_j\gamma$ where $j > i$

$$A_1 \to A_2A_3$$
$$A_2 \to A_3A_1 \mid b$$
$$A_3 \to a$$

(ii) Productions of the form $A_i \to A_j\gamma$ where $j \le i$

$$A_3 \to A_1A_2$$

Step 4. Apply left factoring an production

$A_3 \to A_1A_2$, we get

$A_3 \to A_2A_3A_2$ $\quad (A_1 \to A_2A_3)$

Again apply left factoring an above production

$A_3 \to A_3A_1A_3A_2$ $\quad (A_2 \to A_3A_1)$

$A_3 \to bA_3A_2$ $\quad (A_2 \to b)$

Now, A_3-productions are

$$A_3 \to A_3A_1A_3A_2$$
$$A_3 \to bA_3A_2$$
$$A_3 \to a$$

Step 5. Apply left recursion on A_n-productions :

We introduce new variable Z_3 corresponding to the variable A_3, we get

(i) A_3-productions are :

$$A_3 \to bA_3A_2$$
$$A_3 \to a$$
$$A_3 \to bA_3A_2Z_3$$
$$A_3 \to aZ_3$$

(ii) Z_3-productions are :

$$Z_3 \to A_1A_3A_2$$
$$Z_3 \to A_1A_3A_2Z_3$$

Step 6. A_n-production :

(i) A_3-productions are :

$$A_3 \to bA_3A_2 \mid bA_3A_2Z_3 \mid a \mid aZ_3 \quad \ldots(1)$$

(ii) A_2-productions are :

$$A_2 \to A_3A_2 \mid b$$

Among these productions, we retain $A_2 \to b$ and eliminate $A_2 \to A_3A_1$ by using left factoring, we get,

$$A_2 \to bA_3A_2A_1 \mid bA_3A_2Z_3A_1 \mid aA_1 \mid aZ_3A_1 \quad \ldots(2)$$

(iii) A_1-productions are :

$$A_1 \to A_2A_3$$

Apply left factoring on above production, we get,

$$A_1 \to bA_3A_2A_1A_3 \mid bA_3A_2Z_3A_1A_3 \mid aA_1A_3 \mid aZ_3A_1A_3 \quad \ldots(3)$$

Step 7. Z-productions :

Modify the Z-production using left factoring, we get,

$$Z_3 \to bA_3A_2A_1A_3A_3A_2 / bA_3A_2Z_3A_1A_3A_3A_2 / aA_1A_3A_3A_2 / aZ_3A_1A_3A_3A_2$$

and $$Z_3 \to bA_3A_2A_1A_3A_3A_2Z_3 / bA_3A_2Z_3A_1A_3A_3A_2Z_3 / aA_1A_3A_3A_2Z_3 / aZ_3A_1A_3A_3A_2Z_3 \quad \ldots(4)$$

The productions obtained in equation (1) to (4) constitute the productions of resultant grammar.

4.7 Pumping Lemma for CFG

Pumping lemma is used to prove that certain languages are not context-free.

It states that for infinite context free language L, there exist an integer m such that for any string $W \in L, |W| \geq m$, we can write $W = uvxyz$ with lengths $|vxy| \leq m$ and $|vy| \geq 1$ and it must be :

$$uv^ixy^iz \in L, \forall i \geq 0$$

In other words, it can be defined as follows :

Let G be the context free grammar, there exists m such that every string $W \in L(G)$, with $|W| \geq M$, can be decomposed into $W = uvxyz$ satisfying

1. $v \neq \wedge$ or $y \neq \wedge$;
2. $|vxy| \leq m$;
3. $uv^nxy^nz \in L(G) \forall n \geq 0$

Proof : Let $G = (V_N, \Sigma, P, S)$ be the grammar.

$|V| = m > 0$ and that for every rule $A \to X \in P$, $|X| \leq d$ for some $d > 0$.

Select $L = L(G) = a^{m!}$, and let $W \in L$, $|W| \geq m$.

Consider the derivation

$$S \overset{*}{\Rightarrow} W$$

Let T be the parse tree, each internal nodes are labelled by a non-terminal, every leaf by terminal or $\wedge$.

T has atleast $|W| \geq a^{m-1}$ leaves ($\wedge$-labelled leaves to not contribute to $|W|$), which means that one path of length $\geq m + 1$ from the root to the same leaf.

Example 4.36 : Show that the language $L = \{a^n b^n c^n : n \geq 0\}$ is not context free.

Solution :

Given : $L = \{a^n b^n c^n : n \geq 0\}$

We use pumping lemma to show that language L is not context free language.

Assume for contraction that the $L = \{a^n b^n c^n : n \geq 0\}$ is context free.

Let us consider a number m such that $W \in L$ and $|W| \geq m$.

We pick,

$$W = a^m b^m c^m$$

Now, we write

$$W = uvxyz \text{ with lengths } |vxy| \leq m \text{ and } |vy| \geq 1$$

From, the pumping lemma.

$$uv^i xy^i z \in L \qquad \forall i \geq 0$$

Now, we examine all possible cases for string vxy in W.

Case 1 : vxy is within a^m.

u vxy z

aa a .... a aabbb .... bbb ccc ccc

m m m

v and y consists only from a.

Repeating v and y, $k \geq 1$.

u v^2xy^2 z

aa aaaa .... aaaa aabbb .... bbb ccc ccc

m + k m m

From pumping lemma,

$$uv^2xy^2z \in L, \qquad k \geq 1$$

But, $uv^2xy^2z = a^{m+k}b^m c^m \notin L$

$\Downarrow$

Contradiction

Case 2 : vxy is within b^m.

u vxy z

aaa ... aaabbb .... bbbccc ccc

m m m

Similarly, analysing as in with case 1.

Case 3 : vxy is within c^m.

$$\underbrace{aaa \ldots aaabbb \ldots bbbccc \ldots ccc}_{u \quad vxy \quad z}$$
$$\underbrace{\quad}_{m} \quad \underbrace{\quad}_{m} \quad \underbrace{\quad}_{m}$$

Similar analysis with case 1

Case 4 : vxy overlaps among a^m and b^m.

$$\underbrace{aaa \ldots aaabbb \ldots bbbccc \ldots ccc}_{u \quad vxy \quad z}$$
$$\underbrace{\quad}_{m} \quad \underbrace{\quad}_{m} \quad \underbrace{\quad}_{m}$$

Possibility 1 : v contains only a

y contains only b

$$k_1 + k_2 \geq 1$$

$$\underbrace{aaa \ldots aaaaaaa \ldots bbbbbbb \ldots bbbccc \ldots ccc}_{u \quad vx^2y^2 \quad z}$$
$$\underbrace{\quad}_{m+k_1} \quad \underbrace{\quad}_{m+k_2} \quad \underbrace{\quad}_{m}$$

From pumping lemma :

$$uv^2xy^2z \in L, \qquad k_1 + k_2 \geq 1$$

But, $$uv^2xy^2z = a^{m+k_1}b^{m+k_2}c^m \notin L$$

$$\Downarrow$$

CONTRAD'CTION

Possibility 2 : v contains a and b

y contains only b.

$$\underbrace{aaa \ldots aaabbb \ldots bbbccc \ldots ccc}_{u \quad vxy \quad z}$$
$$\underbrace{\quad}_{m} \quad \underbrace{\quad}_{m} \quad \underbrace{\quad}_{m}$$

$$k_1 + k_2 + k \geq 1$$

$$\underbrace{aaa \ldots aaaaabbaabbbbbbbbb \ldots bbbccc \ldots ccc}_{u \quad v^2xy^2 \quad z}$$
$$\underbrace{\quad}_{m} \quad \underbrace{\quad}_{k_1k_1} \quad \underbrace{\quad}_{m} \quad \underbrace{\quad}_{m}$$

From pumping lemma

$$uv^2xy^2z \in L; \qquad k_1 + k_2 + k \geq 1$$

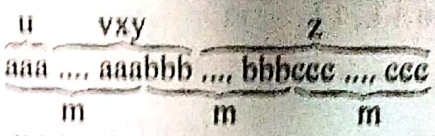
But, $$uv^2xy^2z = a^m b^{k_1} a^{k_2} b^{m+k} c^m \notin L$$

$$\Downarrow$$

CONTRADICTION

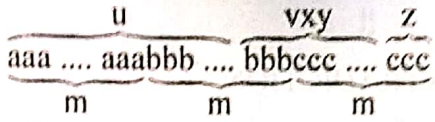
Possibility 3 : v contains only a

y contains a and b,



Similar analysis with possibility 2.

Case 5 : vxy overlaps b^m and c^m.



Similar analysis with case 4.

All the cases have been considered and there are no other cases to consider.

$\because$ $|vxy| \leq m$, string vxy can not overlap a^m, b^m and c^m at the same time.

In all case, we get the CONTRADICTION.

$\therefore$ The assumption that $L = \{a^n b^n c^n : n \geq 0\}$ is context free is wrong.

Conclusion : L is not context free.

Example 4.37 : Prove that the language $L = \{vv : v \in \{a, b\}^*\}$ is not context free.

Solution : Given : $L = \{vv : v \in \{a, b\}^*\}$

Assume for contradiction that language

$L = \{vv : v \in \{a, b\}^*\}$ is context free.

$\because$ L is context free and of infinite length, we can apply punping lemma.

Let us consider a number m such that

$W \in L$ and $|W| \geq m$

We pick, $a^m b^m a^m b^m \in L$

We can write,

$a^m b^m a^m b^m = uvxyz$ with lengths $|vxy| \leq m$ and $|vy| \geq 1$.

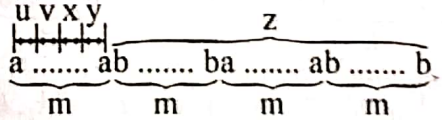
From punping lemma,

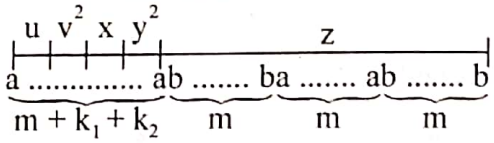
$uv^i xy^i z \in L \quad \forall i \geq 0$

Now, we consider all possible cases : for string vxy in $a^m b^m a^m b^m$.

Case 1 : vxy is within the first a^m :

$v = a^{k_1}, \quad y = a^{k_2}, \quad k_1 + k_2 \geq 1$





$$a^{m+k_1+k_2}b^m a^m b^m = uv^2xy^2z \in L; \qquad k_1 + k_2 \geq 1$$

But, from pumping lemma

$$uv^2xy^2 \in L$$

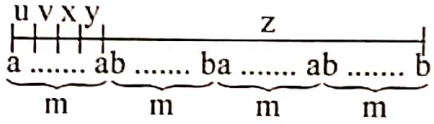
$$\Downarrow$$

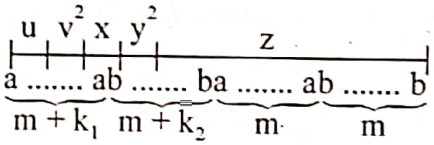
CONTRADICTION

Case 2 : v is in the first a^m

y is in the first b^m

$$v = a^{k_1}, \qquad y = b^{k_2}, \qquad k_1 + k_2 \geq 1$$





$$a^{m+k_1}b^{m+k_2}a^m b^m = uv^2xy^2z \in L; \qquad k_1 + k_2 \geq 1$$

But from pumping lemma.

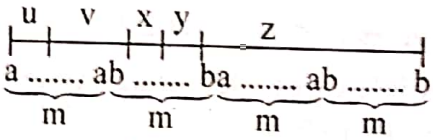
$$uv^2xy^2z \in L$$

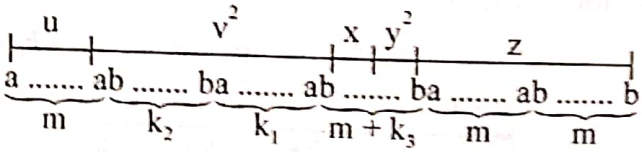
$$\Downarrow$$

CONTRADICTION

Case 3 :

$$v = a^{k_1}b^{k_2}, \qquad y = b^{k_3}, \qquad k_1, k_2 \geq 1$$





$$a^m b^{k_2} a^{k_1} b^{m+k_3} a^m b^m = uv^2xy^2z \in L; \qquad k_1, k_2 \geq 1$$

$$a^m b^{k_2} a^{k_1} b^{k_3} a^m b^m = uv^2xy^2z \notin L$$

But, from pumping lemma.

$$uv^2xy^2z \in L$$

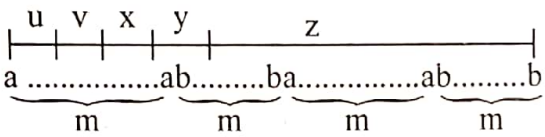
$$\Downarrow$$

CONTRADICTION

Case 4 : v is in first a^m

y overlaps the first $a^m b^m$.

Analysis is similar to case 3.



Other cases : (i) vxy is within $a^m b^m a^m b^m$ means vxy is in first b^m, or second a^m or in second b^m.

Analysis is similar to case 1

i.e. $\underline{a}^m b^m a^m b^m$

(ii) vxy overlaps $a^m \underline{b^m a^m} b^m$ or $a^m b^m \underline{a^m b^m}$

Analysis is similar to case 2, 3 and 4.

There are no other case to consider

$\because$ $|vxy| \leq m$, it is not possible from vxy to overlap : $\underline{a^m b^m} a^m b^m$ nor $a^m \underline{b^m a^m b^m}$ nor $\underline{a^m b^m a^m b^m}$.

In all cases, we get contradiction,

$\therefore$ Our assumption that the language $L = \{vv : v \in \{a, b\}^*\}$ is context free is **not** true.

Conclusion : L is not context free language

Example 4.38 : Prove that the language $L = \{a^{n!} : n \geq 0\}$ is not context free.

Solution : Given : $L = \{a^{n!} : n \geq 0\}$

Assume for contradiction that the language $L = \{a^{n!} : n \geq 0\}$ is context free.

$\because$ The language L is context free and infinite, we can apply the pumping lemma.

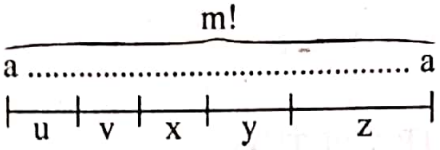
Let us consider a number **m** such that $W \in L$ and $|W| \geq m$.

We pick, $a^{m!} \in L$

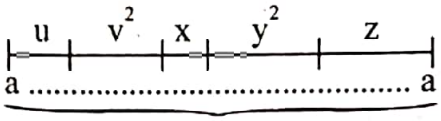
We can write, $a^{m!} = uvxyz$ with lengths $|vxy| \leq m$ and $|vy| \geq 1$

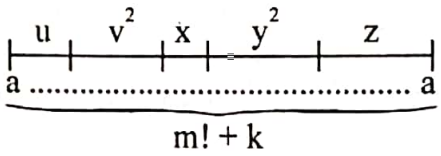
From pumping lemma, $uv^i xy^i z \in L \qquad \forall i \geq 0$

Now, we examine all possible cases of string vxy in $a^{m!}$



$$v = a^{k_1}, \qquad y = a^{k_2} \; ; \qquad 1 \leq k_1 + k_2 \leq m$$





where, $\qquad k = k_1 + k_2; \qquad 1 \leq k \leq m$

$$a^{m!+k} = uv^2xy^2z \qquad (1 \leq k \leq m)$$

$\because$ $\qquad 1 \leq k \leq m, \qquad \forall m \geq 2$

We have,

$$\begin{aligned} m! + k &\leq m! + m \\ &< m! + m!m \\ &= m!(1 + m) \\ &= (m + 1)! \end{aligned}$$

$$\Downarrow$$

$$m! < m! + k < (m + 1)!$$

$$\Downarrow$$

$$a^{m!+k} = uv^2xy^2z \in L$$

However, from pumping lemma

$$vu^2xy^2z \in L$$

$$a^{m!+k} = uv^2xy^2z \in L$$

$$\Downarrow$$

CONTRADICTION

∴ Our assumption that the language $L = \{a^{n!} : n \geq 0\}$ is context free is not true.

Conclusion : L is not context free.

Example 4.39 : Prove that the language $L = \{a^{n^2} b^n : n \geq 0\}$ is not context free.

Solution : Given : $L = \{a^{n^2} b^n : n \geq 0\}$

Assume for contradiction that the the language $L = \{a^{n^2} b^n : n \geq 0\}$ is context free.

∵ L is context free and infinite, we can apply the pumping lemma.

Let us consider the number **m** such that $W \in L$ and $|W| \geq m$.

We pick,

$$a^{m^2} b^m \in L$$

We can write, $a^{m^2 b^m}$ = uvxyz with lengths $|vxy| \leq m$ and $|vy| \geq 1$

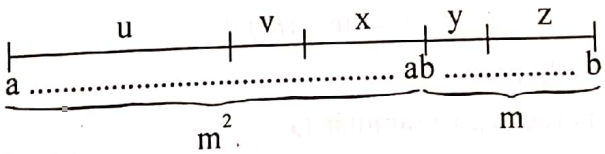
From pumping lemma

$$uv^i xy^i z \in L, \quad \forall i \geq 0$$

Now, we can examine all possible cases of string vxy in $a^{m^2} b^m$.

Case 1 : v is in a^m

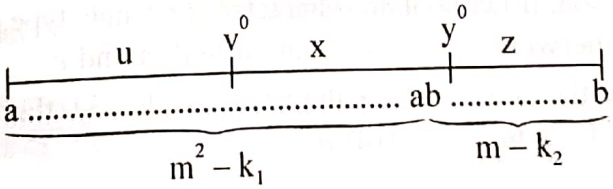
y is in b^m



where $v = a^{k_1}$, $y = b^{k_2}$ and

$$1 \leq k_1 + k_2 \leq m$$

(i) $k \neq 0$ and $k_2 \neq 0$



$$a^{m^2 - k_1} b^{m - k_2} = uv^0 xy^0 z$$

$$k_1 \neq 0 \text{ and } k_2 \neq 0 ; \quad 1 \leq k_1 + k_2 \leq m$$

$$\Downarrow$$

$$(m - k_2)^2 \leq (m - 1)^2$$

$$= m^2 - 2m + 1$$

$$< m^2 - k_1$$

$$\Downarrow$$

$$m^2 - k_1 \neq (m - k_2)^2$$

$$\Downarrow$$

$$a^{m^2-k_1} b^{m-k_2} = uv^0xy^0z \notin L$$

But, from pumping lemma

$$uv^0xy^0z \in L$$

$$a^{m^2-k_1} b^{m-k_2} = uv^0xy^0z \notin L$$

$$\Downarrow$$

CONTRADICTION

$\therefore$ Our assumption that the language $L = \left\{a^{n^2} b^n : n \geq 0\right\}$ is context free is not true.

Conclusion : L is not context free.

Example 4.40 : Show that the set $L = \{a^i b^j c^k \mid k = \max(i, j)\}$ is not context free.

Solution : Given $L = \{a^i b^j c^k \mid k = \max(i, j)\}$

We prove it by contradiction

Assume L is context free with grammar G.

Let $W \in L$ and $|W| \geq m$;

Let $W = a^m b^m c^m$

By pumping lemma

$W = uvxyz$ satisfying the three conditions.

By the length condition, if vxy contains character of a single type, we are done, by 'pumping down or pumping up'. Otherwise, it can not contain both a and c.

1.vy contains c, then the number of c's in uxz is less than m (there are m of them altogether in w), while the number of a's in uxz is still m.

$$\Downarrow$$

CONTRADICTION

2.vy does not contain c. In this case, "pumping up" implies that either the number of a's or that of b's can be increased without altering the number of c's.

$$\Downarrow$$

CONTRADICTION

CFL closed → union, concatenation, star
CFL Not closed → intersection, complement

∴ Our assumption that the language $L = \{a^i b^j c^k \mid k = \max(i, j)\}$ is not true.

Conclusion : L is not context free.

4.8 Properties of CFG

1. Context free languages are closed under **union.**

$$\left.\begin{array}{l} L_1 \text{ is context free} \\ L_2 \text{ is context free} \end{array}\right\} \Rightarrow L_1 \cup L_2 \text{ is context free.}$$

2. Context free languages are closed under **concatanation.**

$$\left.\begin{array}{l} L_1 \text{ is context free} \\ L_2 \text{ is context free} \end{array}\right\} \Rightarrow L_1 L_2 \text{ is context free.}$$

3. Context free languages are closed under **star operation.**

$$L \text{ is context free} \Rightarrow L^* \text{ is context free.}$$

4. Context free languages are **not** closed under intersection.

$$\left.\begin{array}{l} L_1 \text{ is context free} \\ L_2 \text{ is context free} \end{array}\right\} \Rightarrow L_1 \cap L_2 \text{ is } \textbf{not necessarily} \text{ context free.}$$

5. Context free languages are **not** closed under complement.

$$L \text{ is context free} \Rightarrow \overline{L} \text{ is } \textbf{not necessarily} \text{ context free.}$$

6. The **intersection** of a **context free language** and a **regular language** is a **context free language.**

$$\left.\begin{array}{l} L_1 \text{ is context free} \\ L_2 \text{ is regular} \end{array}\right\} \Rightarrow L_1 \cap L_2 \text{ is context free.}$$

4.9 Applications of Context Free Grammar

Context-free grammars are a more powerful method of describing languages compared to regular expressions. They allow recursive structure and were first used in the study of human languages. More modern examples include the structures of mark-up languages like HTML and XML. Another important application of context-free grammars occurs in the specification and compilation of programming languages. Designers of compilers and interpreters for programming languages often start by obtaining a grammar for the language. Context-free grammars improve the process of implementing parsers into a routine job that can be completed very quickly.

Review Questions

1. Construct grammars in GNF generating the set $\{\omega c\omega^T \mid \omega \in \{a, b\}^*\}$.
2. Show that the grammar $S \to aB \mid ab$, $A \to aAB \mid a$, $B \to ABb \mid b$ is ambiguous.
3. Design a Context Free Grammar for generating alternating sequence of 0's and 1's.
4. Describe methods for following tasks :
   (i) Elimination of $\wedge$-productions from a CFG.
   (ii) Elimination of useless symbols from a CFG.
   (iii) Elimination of unit productions from a CFG.
5. Reduce the following grammar to Chomsky Normal Form :

$$S \to abSb$$
$$S \to a$$
$$S \to aAb$$
$$A \to bS$$
$$A \to aAAb$$

6. Using pumping lemma for CFL, prove that language $L = \{a^n b^n c^{2n} \mid n \geq 1\}$ is not context free language.
7. Find a grammar in Chomsky normal form equivalent to :

$$S \to aAbB, A \to aA \mid a, B \to bB \mid b$$

8. Show that the grammar :

$$S \to a \mid abSb \mid aAb,$$
$$A \to bS \mid aAAb$$

is ambiguous.

9. Convert following grammar to an equivalent GNF grammar :

$$S \to S + T \mid T$$
$$T \to T * F \mid F$$
$$F \to (S) \mid a$$

10. Convert the following grammar into CNF grammar.

$$S \to AACD$$
$$A \to aAb \mid \wedge$$
$$C \to aC \mid a$$
$$D \to aDa \mid bDb \mid \wedge$$

11. Show that every context-free language is accepted by a deterministic linear bounded automata.
12. Define normal forms.
13. Find a grammar in Chomsky Normal form equivalent to the grammar :

$$S \to aAbB,$$

$$A \to aA \mid a$$

$$B \to bB \mid b$$

14. Show that the grammar

$$S \to a \mid abSb \mid aAb, A \to bS \mid aAAb$$

is ambiguous.

15. Eliminate the unit production from the CFG with P given by :

$$S \to Aa \mid B$$

$$B \to A \mid bb$$

$$A \to a \mid bc \mid B.$$

16. Construct a grammar in GNF equivalent to the grammar $S \to AB$, $A \to BS/b$, $B \to SA/a$.
17. Let G be the grammar :

$$S \to aB \mid bA$$

$$A \to a \mid aS \mid bAA$$

$$B \to b \mid bS \mid aBB$$

For the string aaabbabbba, find :

(i) Leftmost derivation;

(ii) Rightmost derivation;

(iii) Parse tree.

18. What do you meant by normal form of CFG?
19. When a grammar is said to be ambiguous?
20. Show that the grammer $S \to aB/ab$, $A \to aAB/a$, $B \to ABb/b$ is ambiguous.
21. Consider the following productions

$$S \to aB \mid bA$$

$$A \to aS \mid bAA \mid a$$

$$B \to bS \mid aBB \mid b$$

For the string *aaabbabba*, find a leftmost derivation.

22. Show that the following grammar is ambiguous.

$$S \to a$$
$$S \to abSb$$
$$S \to aAb$$
$$A \to bS$$
$$A \to aAAb$$

23. Eliminate $\wedge$-productions from following grammars.

(*i*)
$$S \to AB \mid \wedge$$
$$A \to aASb \mid a$$
$$B \to bS$$

(*ii*)
$$S \to ABA$$
$$A \to a \mid \wedge$$
$$B \to b \mid \wedge$$

24. Begin with the grammar

$$S \to 0A0 \mid 1B1 \mid BB$$
$$A \to C$$
$$B \to S \mid A$$
$$C \to S \mid \wedge$$

(*i*) Eliminate $\wedge$-productions.

(*ii*) Eliminate unit production.

(*iii*) Eliminate useless symbols.

(*iv*) Put the grammar in Chomsky's normal form.

25. Prove that CFG's are not closed under intersection.