1. The symbol '→' is replaced by the symbol ':
2. All the productions with a given head (i.e., the non-terminals towards left o head) are grouped together and the body of the corresponding productions by vertical bars and ends with terminator ';'
3. The terminals are enclosed within single quotes and the variables are not enclosed within single quotes.

Markup languages: The most familiar markup language is HTML ((Hyper Text Markup Language). HTML is used to create Hypertext documents for use on the World Wide Web. In HTML a block of text is surrounded with codes that indicate how it should appear on the computer screen. In HTML we can specify that a block of text, or a word, is linked to another file on the Internet. Hypertext Markup Language is the code used to write most documents on the World Wide Web. We can write the code using any text editor.

XML: XML stands for Extensible Markup Language. XML is a system for defining, validating, and sharing document formats. XML uses **tags** to distinguish document structures, and **attributes**. Instead of concentrating on formatting the text, XML is used to define the semantics.

Exercises:

1. What are the limitations of regular languages?
2. What is a context free grammar? Explain with example.
3. Let G = (V,T,P,S) be a CFG where

   V = { S }
   T = {a, b}
   P = {
   S → aSa | bSb | ε
   }
   S is the start symbol.
   What is the language generated by this grammar?

4. Show that the language L = { $a^m b^n$ | $m \neq n$ } is context free.
5. Draw a CFG to generate a language consisting of equal number of a's and b's
6. Draw a CFG on {a, b} to generate a language L = { $a^n w w^R b^n$ | $w \in \Sigma^*, n \geq 1$} .
7. Obtain a context free grammar to generate properly nested parentheses structures involving three kinds of parentheses ( ), [ ] and { }.
8. Obtain a context free grammar to generate the following language
   L = {w | $w \in$ {a, b}*, $n_a(v) \geq n_b(v)$ where v is any prefix of w}
9. Obtain a context free grammar to generate the following language
   L = {$01(1100)^n 110(10)^n$ | $n \geq 0$}
10. Is the following language Context free?
    L = {$a^n b^n$ | $n \geq 0$}
11. Obtain a context free grammar to generate the following language
    L = {$a^n b^m$: $m > n$ and $n \geq 0$}

12. Obtain a CFG to generate unequal number of a's and b's
13. For the regular expression (011+1)*(01)* obtain the context free grammar.
14. What is leftmost derivation? Explain with example.
15. What is rightmost derivation? Explain with example.
16. What is a derivation tree (or parse tree)? Explain with example
17. What is the *yield* of a tree? Explain with example
18. What is partial parse tree (or partial derivation tree)? Explain with example.
19. What is an ambiguous grammar?
20. Consider the grammar shown below from which any arithmetic expression can be obtained.

$$\begin{aligned} E &\rightarrow E+E \\ E &\rightarrow E-E \\ E &\rightarrow E*E \\ E &\rightarrow E/E \\ E &\rightarrow (E) \mid \mathbf{id} \end{aligned}$$

Show that the grammar is ambiguous.

21. Is the following grammar ambiguous?

$$\begin{aligned} S &\rightarrow aS \mid X \\ X &\rightarrow aX \mid a \end{aligned}$$

22. Is the following grammar ambiguous?

$$\begin{aligned} S &\rightarrow iCtS \mid iCtSeS \mid a \\ C &\rightarrow b \end{aligned}$$

23. Is the grammar ambiguous?

$$\begin{aligned} S &\rightarrow AB \mid aaB \\ A &\rightarrow a \mid Aa \\ B &\rightarrow b \end{aligned}$$

24. Show that the following grammar is ambiguous

$$\begin{aligned} S &\rightarrow aSbS \\ S &\rightarrow bSaS \\ S &\rightarrow \varepsilon \end{aligned}$$

25. Obtain the unambiguous grammar for the grammar shown

$$\begin{aligned} E &\rightarrow E+E \mid E-E \\ E &\rightarrow E*E \mid E/E \\ E &\rightarrow (E) \mid I \\ I &\rightarrow a \mid b \mid c \end{aligned}$$

and obtain the derivation for the expression (a+b) * (a-b)

26. What is inherently ambiguous grammar?
27. What is an S-Grammar (Simple Grammar)? Explain with example.

28. Find a Simple Grammar (S-Grammar) for the regular expression aaa*b + b
29. Find a Simple Grammar (S-Grammar) to generate the language $L = \{a^n b^n \mid n \geq 1\}$
30. In what way BNF notations are different from the usual representation of the grammar? Give an example.
31. Use BNF notation and describe the while statement in C language. Assume that assignment statement and condition for while are defined already.
32. Give the BNF notation to write a C program. Provide 6 or 7 productions generally describing the main program. Assume the rest are defined.
33. Obtain a derivation tree for the string a + b * a + b from the grammar shown below:

$$\begin{aligned} E &\rightarrow E+E \\ E &\rightarrow E-E \\ E &\rightarrow E*E \\ E &\rightarrow E/E \\ E &\rightarrow a \mid b \end{aligned}$$

34. Consider the grammar :

$$\begin{aligned} S &\rightarrow 0 \mid 01S1 \mid 0A1 \\ A &\rightarrow 1S \mid 0AA1 \end{aligned}$$

Is the grammar ambiguous? Ans: yes

35. What language is accepted by the following grammars?
    a. $S \rightarrow 0S0 \mid 1S1 \mid \varepsilon$
    b. $S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1$
    c. $S \rightarrow 0S1 \mid 1S1 \mid \varepsilon$
       $A \rightarrow 0B1 \mid 1B0$
       $B \rightarrow 0B0 \mid 1B1 \mid 0 \mid 1 \mid \varepsilon$
    d. $S \rightarrow cS \mid cSdS \mid \varepsilon$
    e. $S \rightarrow cS \mid dS \mid c$
    f. $S \rightarrow SS \mid dS \mid Sd \mid c$

36. Is the grammar $S \rightarrow SS \mid (S) \mid \varepsilon$ ambiguous? (Ans : Yes) obtain two derivations by applying LMD or obtain two derivations by applying RMD.

Summary:

Now! We know

- Advantages of regular and non-regular languages
- Definition of Context Free Grammar (CFG)
- To obtain CFGs for various types of context free languages
- To check whether the given languages is CFG
- Leftmost derivation
- Rightmost derivation
- Derivation Tree(Parse tree)
- Yield of a tree
- Partial derivation tree
- Ambiguous grammar
- Solution to various problems of ambiguous grammars
- Inherently ambiguous grammar
- Parsing
- Simple grammar (S-Grammar)
- To obtain S-Grammars for the specified languages
- Applications of context free grammars
- BNF notations with applications
- Solution to more than 30 problems of various nature

6

CFG Simplification & Normal Forms

What we will know after reading this chapter?

- Method of substitution
- Left recursion
- Procedure to eliminate left recursion
- Simplification of the grammar with proof
- Solutions to simplify various types of grammars
- Useless variable
- ε - Production
- Nullable variable
- Elimination of ε - productions
- Unit production
- Elimination of unit productions
- Chomsky Normal Form (CNF)
- Conversion of various types of grammars to CNF notation
- Greiback Normal Form (GNF)
- Conversion of various types of grammars to GNF notations
- Solution to more than 15 problems of various nature

Even though there is no restriction on the right hand side of the production for any CFG, it is better in fact necessary to eliminate some of the useless symbols and productions. In the grammar G, some of the symbols or productions may not be used to derive a string. Some symbols and productions may never be used while deriving a string. So, these symbols and productions which will never be used are useless and the corresponding productions can be eliminated. For example, consider the grammar

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

In this grammar if we apply the production S → B, a string can never be derived. So, the symbol B and the production S → B are useless and can be eliminated. In the following sections, we discuss how to eliminate

- symbols in V from which string of terminals can not be derived
- symbols in (V U T) and not appearing in any sentential form
- ε- productions
- The productions of the form A → B i.e., unit productions

Thus, a CFG can be simplified by eliminating ε-productions, useless symbols, unit productions etc. This chapter covers the simplification process and two normal forms: Chmosky normal form and Greibach normal form.

6.1 Substitution

The section 6.1 and 6.2 provides different substitution methods. This section deals with a simple substitution wherein a non-terminal is replaced by the corresponding symbols on the right hand side.

Theorem 6.1: Let G = (V, T, P, S) be a context free grammar. Consider the productions

$$A \rightarrow x_1Bx_2$$

and

$$B \rightarrow y_1 | y_2 | - - - y_n$$

The production

$$A \rightarrow x_1Bx_2$$

can be replaced by

$$A \rightarrow x_1y_1x_2 | x_1y_2x_2 | \ldots | x_1y_nx_2$$

and the production

$$B \rightarrow y_1 | y_2 | --- y_n$$

in P can be deleted. The resulting productions are added to P_1 and the variables are added to V_1. The language generated by the resulting grammar $G_1 = (V_1, T, P_1, S)$ is same as the language accepted by G i.e., $L(G_1) = L(G)$.

Example 6.1: Consider the productions

$$\begin{aligned} A &\rightarrow aBa \\ B &\rightarrow ab \mid b \end{aligned}$$

Simplify the grammar by substitution method.

Consider the production

$$A \rightarrow aBa$$

The right hand side of the production contains a non terminal B. The non-terminal B can be replaced by the production

$$B \rightarrow ab \mid b$$

as shown below:

$$A \rightarrow aaba \mid aba$$

So, the resulting grammar is G = (V, T, P, S) where

V = {A}
T = {a, b}
P = {A → aaba | aba}
A is the start symbol.

6.2 Left Recursion

A grammar can be changed from one form to another accepting the same language. Another important substitution method which is often useful is left recursion. If a grammar has left recursive property, it is undesirable as the parser constructed from this left recursive grammar will enter into an infinite loop and the system may crash. For this reason left recursion should be eliminated from the grammar. The left recursion is defined as follows.

Definition: A grammar G is said to be left recursive if there is some non-terminal A such that

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

In other words, if the first symbol on the right hand side in a sentential form (either left sentential or right sentential) is a variable and if the derivation is obtained from the same non-terminal, then we say that the grammar is having left recursion.

The left recursion in a grammar G can be eliminated as shown below.

Consider the A–production of the form

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \ldots\ldots A\alpha_n \mid \beta_1 \mid \beta_2 \mid \beta_3 \ldots\ldots \beta_m$$

where β_i's do not start with A. Then the A productions can be replaced by

$$A \rightarrow \beta_1 A^1 \mid \beta_2 A^1 \mid \beta_3 A^1 \mid \ldots\ldots\ldots \beta_m A^1$$
$$A^1 \rightarrow \alpha_1 A^1 \mid \alpha_2 A^1 \mid \alpha_3 A^1 \mid \ldots\ldots\ldots \mid \alpha_n A^1 \mid \varepsilon$$

Note that α_i's do not start with A^1.

Example 6.2: Eliminate left recursion from the following grammar

$$\begin{aligned} E &\rightarrow E+T \mid T \\ T &\rightarrow T*F \mid F \\ F &\rightarrow (E) \mid id \end{aligned}$$

The left recursion can be eliminated as shown below:

Given $A \rightarrow A\alpha_i \mid \beta_i$	Substitution	Without left recursion $A \rightarrow \beta_i A'$ and $A' \rightarrow \alpha_i A' \mid \varepsilon$
$E \rightarrow E+T \mid T$	$A = E$ $\alpha_1 = +T$ $\beta_1 = T$	$E \rightarrow TE'$ $E' \rightarrow +TE' \mid \varepsilon$
$T \rightarrow T*F \mid F$	$A = T$ $\alpha_1 = *F$ $\beta_1 = F$	$T \rightarrow FT'$ $T' \rightarrow *FT' \mid \varepsilon$
$F \rightarrow (E) \mid id$	Not applicable	$F \rightarrow (E) \mid id$

The grammar obtained after eliminating left recursion is

$$\begin{aligned} E &\rightarrow TE' \\ E' &\rightarrow +TE' \mid \varepsilon \\ T &\rightarrow FT' \\ T' &\rightarrow *FT' \mid \varepsilon \\ F &\rightarrow (E) \mid id \end{aligned}$$

Example 6.3: Eliminate left recursion from the following grammar

$$\begin{aligned} S &\rightarrow Ab \mid a \\ A &\rightarrow Ab \mid Sa \end{aligned}$$

The non terminal S, even though is not having immediate left recursion, it has left recursion since

$$S \Rightarrow Ab \Rightarrow Sab$$

i.e.,

$$S \overset{+}{\Rightarrow} Sab.$$

Substituting for S in the A-production can eliminate the indirect left recursion from S. So, the given grammar can be written as

$$\begin{aligned} S &\rightarrow Ab \mid a \\ A &\rightarrow Ab \mid Aba \mid aa \end{aligned}$$

Now, A-production has left recursion and can be eliminated as shown below:

Given $A \rightarrow A\alpha_i \mid \beta_i$	Substitution	Without left recursion $A \rightarrow \beta_i A' \text{ and } A' \rightarrow \alpha_i A' \mid \varepsilon$
$S \rightarrow Ab \mid a$	Not applicable	$S \rightarrow Ab \mid a$
$A \rightarrow Ab \mid Aba \mid aa$	$A = A$ $\alpha_1 = b$ $\alpha_2 = ba$ $\beta_1 = aa$	$A \rightarrow aaA'$ $A' \rightarrow bA' \mid baA' \mid \varepsilon$

The grammar obtained after eliminating left recursion is

$$\begin{aligned} S &\rightarrow Ab \mid a \\ A &\rightarrow aaA' \\ A' &\rightarrow bA' \mid baA' \mid \varepsilon \end{aligned}$$

6.3 Simplification of CFG

In a CFG, it may be necessary to eliminate some of the useless symbols and productions. Let G = (V, T, P, S) be a CFG. In the grammar G, some of the symbols or productions may not be used to derive a string and some symbols and productions may not be reachable from the start symbol. So, these symbols and productions which are not used in any sentential from are useless and the corresponding productions can be eliminated. For example, consider the grammar

$$\begin{aligned} S &\rightarrow aA \mid B \\ A &\rightarrow aA \mid a \end{aligned}$$

In this grammar if we apply the production $S \rightarrow B$, from B, a string can never be derived. So, the symbol B and the production $S \rightarrow B$ are useless and can be eliminated. So, in this section, let us concentrate on how a grammar can be simplified by eliminating useless symbols and variables.

Theorem 6.2: Let G = (V, T, P, S) be a CFG. We can find an equivalent grammar $G' = (V', T', P', S)$ such that for each A in $(V' \cup T')$ there exists α and β in $(V' \cup T')^*$ and x in T^+ for which

$$S \overset{*}{\Rightarrow} \alpha A \beta \overset{*}{\Rightarrow} x.$$

Note: It means that any variable or symbols which are not reachable from the start symbol and which are not used while deriving a string of terminals the symbols are useless and all productions which contains those symbols are also useless.

Proof: The grammar G' can be obtained from G in two steps.

Stage 1: Obtain the set of variables and productions which derive only string of terminals i.e., Obtain a grammar $G_1 = (V_1, T_1, P_1, S)$ such that V_1 contains only the set of variables A for which

$$A \overset{*}{\Rightarrow} x$$

where $x \in T^+$. The algorithm to obtain a set of variables from which only string of terminals can be derived is shown below.

Step 1: [Initialize old_variables denoted by *ov* to ϕ]

$ov = \phi$

Step 2: Take all productions of the form $A \rightarrow x$ where $x \in T^+$ i.e., if the R.H.S of the production contains only string of terminals consider those productions and corresponding non terminals on L.H.S are added to new_variables denoted by *nv*. This can be expressed using the following statement:

$nv = \{A \mid A \rightarrow x \text{ and } x \in T^+\}$

Step 3: Compare *ov* and *nv*. As long as the elements in *ov* and *nv* are not equal, repeat the following statements. Otherwise go to step 4.

a. [Copy new_variables to old_variables]

$ov = nv$

b. Add all the elements in *ov* to *nv*. Also add the variables which derive a string consisting of terminals and non-terminals which are in *ov* i.e.,

$nv = ov \cup \{A \mid A \rightarrow y \text{ and } y \in (ov \cup T)^*\}$

The step 3 can be written in algorithmic notation as

```
while ( ov != nv )
{
    ov = nv;
    nv = ov U{A | A → y  and y ∈(ov U T)* }
}
```

Step 4: When the loop is terminated, nv(or *ov*) contains all those non-terminals from which only the string of terminals are derived and add those variables to V_1.

i.e., $V_1 = ov$

Step 5: [Terminate the algorithm]

return V_1

The complete algorithm is shown below:

```
ov = φ
nv = ov U{A | A → y  and y ∈(ov U T)* }
```

```
while ( ov != nv )
{
    ov = nv;
    nv = ov U{A | A → y  and y ∈(ov U T)* }
}
V1 = ov
```

Note that the variable V_1 contains only those variables from which string of terminals are obtained. The productions used to obtain V_1 are added to P_1 and the terminals in these productions are added to T_1. The grammar $G_1 = (V_1, T_1, P_1, S)$ contains those variables A in V_1 such that

$$A \stackrel{*}{\Rightarrow} x$$

for some x in T^+. Since each derivation in G_1 is a derivation of G,

$$L(G_1) = L(G).$$

Stage 2: Obtain the set of variables and terminals which are reachable from the start symbol. The productions which are not used are useless. This can be obtained as shown below:

Given a CFG $G_1 = (V_1,T_1,P_1,S)$, we can find an equivalent grammar $G' = (V',T',P',S)$ such that for each X in $(V' \cup T')$ there exists some α such that

$$S \stackrel{*}{\Rightarrow} \alpha$$

where X is a symbol in α i.e., if X is a variable, $X \in V'$ and if X is a terminal $X \in T'$. Each symbol X in $(V' \cup T')$ is reachable from the start symbol S. The algorithm for this is shown below.

```
V' = {S}
For each A in V'
    If A → α then
            Add the variables in α to V'
            Add the terminals in α to T'
    Endif
Endfor
```

Using this algorithm all those symbols (whether variables or terminals) that are not reachable from the start symbol are eliminated. The grammar G' does not contain any useless symbol or production. For each $x \in L(G')$ there is a derivation

$$S \stackrel{*}{\Rightarrow} \alpha X \beta \stackrel{*}{\Rightarrow} x$$

Using these two steps we can effectively find G' such that L(G) = L(G') and the two grammars G and G' are equivalent.

Definition: A symbol X is useful if there is a derivation of the form

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$$

Otherwise, the symbol X is useless. Note that in a derivation, finally we should get string of terminals and all these symbols must be reachable from the start symbol S. Those symbols and productions which are not at all used in the derivation are useless.

Example 6.4: Eliminate the useless symbols in the grammar

$$\begin{aligned} S &\rightarrow aA \mid bB \\ A &\rightarrow aA \mid a \\ B &\rightarrow bB \\ D &\rightarrow ab \mid Ea \\ E &\rightarrow aC \mid d \end{aligned}$$

Stage 1 : Applying the algorithm shown in stage1, we can obtain a set of variables from which we get only string of terminals and is shown below.

ov	nv	Productions
ϕ	A, D, E	A → a D → ab E → d
A,D,E	A,D,E,S	S → aA A → aA D → Ea
A,D,E,S	A,D,E,S	-

The resulting grammar $G_1 = (V_1, T_1, P_1, S)$ where

V_1 = {A,D,E,S}
T_1 = {a, b, d}
P_1 = {

$$\begin{aligned} A &\rightarrow a \mid aA \\ D &\rightarrow ab \mid Ea \\ E &\rightarrow d \\ S &\rightarrow aA \end{aligned}$$

}
S is the start symbol

contains all those variables in V_1 such that $A \overset{+}{\Rightarrow} w$ where $w \in T^+$.

Stage 2: Applying the algorithm given in stage 2 of the theorem 6.2, we obtain the symbols such that each symbol X is reachable from the start symbol S as shown below.

P^1	T^1	V^1
-	-	S
$S \rightarrow aA$	a	S, A
$A \rightarrow a \mid aA$	a	S, A

The resulting grammar $G^1 = (V^1, T^1, P^1, S)$ where

V^1 = { S, A }
T^1 = {a}
P^1 = {
S → aA
A → a | aA
}
S is the start symbol

such that each symbol X in $(V^1 \cup T^1)$ has a derivation of the form

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w.$$

Example 6.5: Simplify the following grammar

S → aA | a | Bb | cC
A → aB
B → a | Aa
C → cCD
D → ddd

Stage 1 : Applying the algorithm shown in stage1 of theorem 6.2, we can obtain a set of variables from which we get only string of terminals and is shown below.

ov	nv	Productions
ϕ	S, B, D	S → a B → a D → ddd
S, B, D	S, B, D, A	S → Bb A → aB
S, B, D, A	S, B, D, A	S → aA B → Aa

The resulting grammar $G_1 = (V_1, T_1, P_1, S)$ where

V_1 = {S, B, D, A}
T_1 = {a, b, d}

P_1 = {
S → a | Bb | aA
B → a | Aa
D → ddd
A → aB
}
S is the start symbol

contains all those variables in V_1 such that $A \overset{+}{\Rightarrow} w$.

Stage 2: Applying the algorithm given in stage 2 of the theorem 6.2, we obtain the symbols such that each symbol X is reachable from the start symbol S as shown below.

P^1	T^1	V^1
-	-	S
S → a | Bb | Aa	a, b	S, A, B
A → aB	a, b	S, A, B
B → a | Aa	a, b,	S, A, B

The resulting grammar $G^1 = (V^1, T^1, P^1, S)$ where

V^1 = { S, A, B }
T^1 = {a, b}
P^1 = {
S → a | Bb | aA
A → aB
B → a | Aa
}
S is the start symbol

such that each symbol X in $(V^1 \cup T^1)$ has a derivation of the form

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w.$$

6.4 Eliminating ε - productions

A production of the form A → ε is undesirable in a CFG, unless an empty string is derived from the start symbol. Suppose, the language generated from a grammar G does not derive any empty string and the grammar consists of ε-productions. Such ε-productions can be removed. An ε-production is defined as follows:

Definition: Let G = (V, T, P, S) be a CFG. A production in P of the form

$$A \rightarrow \varepsilon$$

is called an ε-production or NULL production. After applying the production the variable A is erased. For each A in V, if there is a derivation of the form

$$A \overset{*}{\Rightarrow} \varepsilon$$

then A is a nullable variable.

Example 6.6: Consider the grammar

$$\begin{aligned} S &\rightarrow ABCa \mid bD \\ A &\rightarrow BC \mid b \\ B &\rightarrow b \mid \varepsilon \\ C &\rightarrow c \mid \varepsilon \\ D &\rightarrow d \end{aligned}$$

In this grammar, the productions

$$B \rightarrow \varepsilon$$
$$C \rightarrow \varepsilon$$

are ε- productions and the variables B, C are nullable variables. Because there is a production

$$A \rightarrow BC$$

and both B and C are nullable variables, then A is also a nullable variable.

Definition: Let G = (V, T, P, S) be a CFG where V set of variables, T is set of terminals, P is set of productions and S is the start symbol. A nullable variable is defined as follows.

1. If $A \rightarrow \varepsilon$ is a production in P, then A is a nullable variable.
2. If $A \rightarrow B_1B_2 \ldots B_n$ is a production in P, and if $B_1, B_2, \ldots B_n$ are nullable variables, then A is also a nullable variable
3. The variables for which there are productions of the form shown in step 1 and step2 are nullable variables.

Even though a grammar G has some ε-productions, the language may not derive a language containing empty string. So, in such cases, the ε- productions or NULL productions are not needed and they can be eliminated.

Theorem 6.3: Let G = (V,T,P,S) where $L(G) \neq \varepsilon$. We can effectively find an equivalent grammar G' with no ∈ - productions such that $L(G') = L(G) - \varepsilon$.

Proof : The grammar G' can be obtained from G in two steps.

Step1: find the set of nullable variables in the grammar G using the following algorithm.

```
ov = ϕ

nv = { A | A → ε }

while ( ov != nv )
{
        ov = nv
        nv = ov ∪ { A | A → α and α ∈ ov* }
}

V = ov
```

Once the control comes out of the while loop, the set V contains only the nullable variables.

Step2: Construction of productions P'. Consider a production of the form

$$A \rightarrow X_1X_2X_3.....X_n \text{ , } n \geq 1$$

where each X_i is in (V U T). In a production, take all possible combinations of nullable variables and replace the nullable variables with ε one by one and add the resulting productions to P'. If the given production is not an ε- production, add it to P'.

Suppose, A and B are nullable variables in the production, then

1. First add the production to P'.
2. Replace A with ε in the given production and add the resulting production to P'.
3. Replace B with ε in the given production and add the resulting production to P'.
4. Replace A and B with ε and add the resulting production to P'.
5. If all symbols on right side of production are nullable variables, the resulting production is an ε- production and do not add this to P'.

Thus, the resulting grammar G' obtained, generates the same language as generated by G without ε and the proof is straight forward.

Example 6.7: Eliminate all ε- productions from the grammar

S → ABCa | bD
A → BC | b
B → b | ε
C → c | ε
D → d

Step1: Obtain the set of nullable variables from the grammar. This can be done using step 1 of theorem 6.3 as shown below.

ov	nv	Productions
ϕ	B,C	$B \rightarrow \varepsilon$ $C \rightarrow \varepsilon$
B,C	B,C,A	$A \rightarrow BC$
B,C,A	B,C,A	-

V = {B,C,A} are all nullable variables.

Step2: Construction of productions P'.

Productions	Resulting productions (P')
$S \rightarrow ABCa$	$S \rightarrow ABCa \mid BCa \mid ACa \mid ABa \mid Ca \mid Aa \mid Ba \mid a$
$S \rightarrow bD$	$S \rightarrow bD$
$A \rightarrow BC \mid b$	$A \rightarrow BC \mid B \mid C \mid b$
$B \rightarrow b \mid \varepsilon$	$B \rightarrow b$
$C \rightarrow c \mid \varepsilon$	$C \rightarrow c$
$D \rightarrow d$	$D \rightarrow d$

The grammar $G' = (V', T', P', S)$ where

$V^1 = \{S, A, B, C, D\}$
$T^1 = \{a, b, c, d\}$
$P^1 = \{$
$S \rightarrow ABCa \mid BCa \mid ACa \mid ABa \mid Ca \mid Aa \mid Ba \mid a \mid bD$
$A \rightarrow BC \mid B \mid C \mid b$
$B \rightarrow b$
$C \rightarrow c$
$D \rightarrow d$
$\}$
S is the start symbol

Example 6.8: Eliminate all ε- productions from the grammar

$S \rightarrow BAAB$
$A \rightarrow 0A2 \mid 2A0 \mid \varepsilon$
$B \rightarrow AB \mid 1B \mid \varepsilon$

Step1: Obtain the set of nullable variables from the grammar. This can be achieved using step 1 of theorem 6.3 as shown below.

ov	nv	Productions
ϕ	A,B	$A \rightarrow \varepsilon$ $B \rightarrow \varepsilon$
A, B	A, B, S	$S \rightarrow BAAB$
A, B, S	A, B, S	-

V = {S, A. B} are all nullable variables.

Step2: Construction of productions P'. Add a non $\in$-production in P to P'. Take all the combinations of nullable variables in a production, delete subset of nullable variables one by one and add the resulting productions to P'.

Productions	Resulting productions (P')
$S \rightarrow BAAB$	$S \rightarrow BAAB \mid AAB \mid BAB \mid BAA \mid AB \mid BB \mid BA \mid AA \mid A \mid B$
$A \rightarrow 0A2$	$A \rightarrow 0A2 \mid 02$
$A \rightarrow 2A0$	$A \rightarrow 2A0 \mid 20$
$B \rightarrow AB$	$B \rightarrow AB \mid B \mid A$
$B \rightarrow 1B$	$B \rightarrow 1B \mid 1$

We can delete the productions of the form $A \rightarrow A$. In P', the production $B \rightarrow B$ can be deleted and the final grammar obtained after eliminating ε-productions is shown below.

The grammar $G' = (V', T', P', S)$ where

$V' = \{S, A, B\}$
$T' = \{0, 1, 2\}$
$P' = \{$
$S \rightarrow BAAB \mid AAB \mid BAB \mid BAA \mid AB \mid BB \mid BA \mid AA \mid A \mid B$
$A \rightarrow 0A2 \mid 02 \mid 2A0 \mid 20$
$B \rightarrow AB \mid A \mid 1B \mid 1$
$\}$
S is the start symbol

6.5 Eliminating unit productions

Consider the productions $A \rightarrow B$. The left hand side of the production and right hand side of the production contains only one variable. Such productions are called unit productions. Formally, a unit production is defined as follows.

Definition: Let G = (V, T, P, S) be a CFG. Any production in G of the form

$$A \rightarrow B$$

where A, B $\in$ V is a unit-production.

In any grammar, the unit productions are undesirable. This is because one variable is simply replaced by another variable. Consider the productions

$$A \rightarrow B$$
$$B \rightarrow aB \mid b$$

In this example,

$$B \rightarrow aB$$
$$B \rightarrow b$$

are non unit productions. Since B is generated from A, whatever is generated by B, the same things can be generated from A also. So, we can have

$$A \rightarrow aB$$
$$A \rightarrow b$$

and the production $A \rightarrow B$ can be deleted.

Theorem 6.4: Let G = (V, T, P, S) be a CFG and has unit productions and no $\in$-productions. An equivalent grammar G_1 without unit productions can be obtained such that $L(G) = L(G_1)$ i.e., any language generated by G is also generated by G_1. But, the grammar G_1 has no unit productions.

A unit production in grammar G can be eliminated using the following steps:

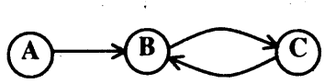
1. Remove all the productions of the form $A \rightarrow A$
2. Add all non unit productions to P_1.
3. For each variable A find all variables B such that

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

   i.e., in the derivation process from A, if we encounter only one variable in a sentential form say B (no terminals should be there), obtain all such variables.
4. Obtain a dependency graph. For example, if we have the productions

   $$A \rightarrow B$$
   $$B \rightarrow C$$
   $$C \rightarrow B$$

   the dependency graph will be of the form



5. Note from the dependency graph that
   a. $A \overset{*}{\Rightarrow} B$ i.e., B can be obtained from A
      So, all non-unit productions generated from B can also be generated from A
   b. $A \overset{*}{\Rightarrow} C$ i.e., C can be obtained from A
      So, all non-unit productions generated from C can also be generated from A
   c. $B \overset{*}{\Rightarrow} C$ i.e., C can be obtained from B
      So, all non-unit productions generated from C can also be generated from B
   d. $C \overset{*}{\Rightarrow} B$ i.e., B can be obtained from C
      So, all non-unit productions generated from B can also be generated from C
6. Finally, the unit productions can be deleted from the grammar G.
7. The resulting grammar G_1, generates the same language as accepted by G.

Example 6.9: Eliminate all unit productions from the grammar

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow a \\ B &\rightarrow C \mid b \\ C &\rightarrow D \\ D &\rightarrow E \mid bC \\ E &\rightarrow d \mid Ab \end{aligned}$$

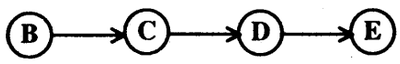
The non unit productions of the grammar G are shown below:

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow a \\ B &\rightarrow b \\ D &\rightarrow bC \\ E &\rightarrow d \mid Ab \end{aligned} \qquad (6.1)$$

The unit productions of the grammar G are shown below:

$$\begin{aligned} B &\rightarrow C \\ C &\rightarrow D \\ D &\rightarrow E \end{aligned}$$

The dependency graph for the unit-productions is shown below:



It is clear from the dependency graph that $D \overset{*}{\Rightarrow} E$. So, all non unit productions generated from E can also be generated from D. The non unit productions from E are

$$E \rightarrow d \mid Ab \qquad (6.2)$$

can also be obtained from D.

$$D \rightarrow d \mid Ab$$

The resulting D productions are

$$\begin{aligned} &D \rightarrow bC \quad \text{(from 6.1)} \\ &D \rightarrow d \mid Ab \end{aligned} \qquad (6.3)$$

From the dependency graph it is clear that, $C \overset{*}{\Rightarrow} E$. So, the non unit productions from E shown in (6.2) can be generated from C. Therefore,

$$C \rightarrow d \mid Ab$$

From the dependency graph it is clear that, $C \overset{*}{\Rightarrow} D$. So, the non unit productions from D shown in (6.3) can be generated from C. Therefore,

$$\begin{aligned} &C \rightarrow bC \\ &C \rightarrow d \mid Ab \end{aligned} \qquad (6.4)$$

From the dependency graph it is clear that $B \overset{*}{\Rightarrow} C$, $B \overset{*}{\Rightarrow} D$, $D \overset{*}{\Rightarrow} E$. So, all the productions obtained from B can be obtained using (6.1), (6.2), (6.3) and (6.4) and the resulting productions are:

$$\begin{aligned} &B \rightarrow b \\ &B \rightarrow d \mid Ab \\ &B \rightarrow bC \end{aligned} \qquad (6.5)$$

The final grammar obtained after eliminating unit productions can be obtained by combining the productions (6.1), (6.2), (6.3), (6.4) and (6.5) and is shown below:

V^1 = {S, A, B, C, D, E}
T^1 = {a, b, d}
P^1 = {

S	→	AB
A	→	a
B	→	b | d | Ab | bC
C	→	bC | d | Ab
D	→	bC | d | Ab
E	→	d | Ab

}
S is the start symbol

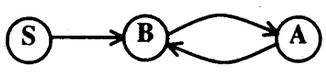
Example 6.10: Eliminate unit productions from the grammar

$$\begin{aligned} S &\rightarrow A0 \mid B \\ B &\rightarrow A \mid 11 \\ A &\rightarrow 0 \mid 12 \mid B \end{aligned}$$

The dependency graph for the unit productions

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

is shown below.



The non unit productions are

$$\begin{aligned} S &\rightarrow A0 \\ B &\rightarrow 11 \\ A &\rightarrow 0 \mid 12 \end{aligned} \tag{6.6}$$

It is clear from the dependency graph that $S \overset{*}{\Rightarrow} B$, $S \overset{*}{\Rightarrow} A$, $B \overset{*}{\Rightarrow} A$ and $A \overset{*}{\Rightarrow} B$. So, the new productions from S, A and B are

$$\begin{aligned} S &\rightarrow 11 \mid 0 \mid 12 \\ B &\rightarrow 0 \mid 12 \\ A &\rightarrow 11 \end{aligned} \tag{6.7}$$

The resulting grammar without unit productions can be obtained by combining (6.6) and (6.7) and is shown below:

$V^1 = \{S, A, B\}$
$T^1 = \{0, 1, 2\}$
$P^1 = \{$
$S \rightarrow A0 \mid 11 \mid 0 \mid 12$
$A \rightarrow 0 \mid 12 \mid 11$
$B \rightarrow 11 \mid 0 \mid 12$
$\}$
S is the start symbol

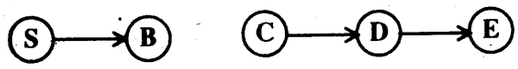
Example 6.11: Eliminate unit productions from the grammar

$S \rightarrow Aa \mid B \mid Ca$
$B \rightarrow aB \mid b$
$C \rightarrow Db \mid D$
$D \rightarrow E \mid d$
$E \rightarrow ab$

The dependency graph for the unit productions

$$S \rightarrow B$$
$$C \rightarrow D$$
$$D \rightarrow E$$

is shown below.



The non unit productions are

$$\begin{aligned} S &\rightarrow Aa \mid Ca \\ B &\rightarrow aB \mid b \\ C &\rightarrow Db \\ D &\rightarrow d \\ E &\rightarrow ab \end{aligned} \tag{6.8}$$

It is clear from the first dependency graph that $S \overset{*}{\Rightarrow} B$ and so whatever is derivable from B it is also derivable from S and the resulting S-productions are:

$$S \rightarrow aB \mid b \tag{6.9}$$

It is clear from the second dependency graph $C \overset{*}{\Rightarrow} D$ and $D \overset{*}{\Rightarrow} E$. So, whatever is derivable from E is also derivable from D and the resulting D-productions are:

$$D \rightarrow ab \mid d \tag{6.10}$$

and the D productions are also derivable from C since $C \overset{*}{\Rightarrow} D$. So, the resulting C-productions are:

$$C \rightarrow Db \mid ab \mid d \tag{6.11}$$

The resulting grammar without unit productions can be obtained by combining (6.8), (6.9), (6.10) and (6.11) and is shown below:

V^1 = {S, A, B, C, D, E}
T^1 = {a, b, d}
P^1 = {

$$\begin{aligned} S &\rightarrow Aa \mid Ca \mid aB \mid b \\ B &\rightarrow aB \mid b \\ C &\rightarrow Db \mid ab \mid d \\ D &\rightarrow d \mid ab \\ E &\rightarrow ab \end{aligned}$$

}
S is the start symbol

Note : Given any grammar, all undesirable productions can be eliminated by removing

1. ε-productions using theorem 6.3
2. unit productions using theorem 6.4
3. useless symbols and productions using theorem 6.2

in sequence. The final grammar obtained does not have any undesirable productions.

6.6 Chomsky Normal Form

In a CFG, there is no restriction on the right hand side of a production. The restrictions are imposed on the right hand side of productions in a CFG resulting in various normal forms. The different normal forms that we discuss are:

1. Chomsky Normal Form (CNF)
2. Greiback Normal Form (GNF)

Chomsky normal form can be defined as follows.

Definition: Let G = (V, T, P, S) be a CFG. The grammar G is said to be in CNF if all productions are of the form

$$A \rightarrow BC$$

or

$$A \rightarrow a$$

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

Note that if a grammar is in CNF, the right hand side of the production should contain two symbols or one symbol. If there are two symbols on the right hand side those two symbols must be non-terminals and if there is only one symbol, that symbol must be a terminal.

Theorem 6.5: Let G = (V, T, P, S) be a CFG which generates context free language without ε. We can find an equivalent context free grammar $G^1 = (V^1, T, P^1, S)$ in CNF such that

$$L(G) = L(G^1)$$

i.e., all productions in G^1 are of the form

$$A \rightarrow BC$$

or

$$A \rightarrow a.$$

Proof: Let the grammar G has no ε-productions and unit productions. The grammar G^1 can be obtained using the following steps.

Step1: Consider the productions of the form

$$A \rightarrow X_1X_2X_3 - - - -X_n$$

where $n \geq 2$ and each $X_i \in (V \cup T)$ i.e., consider the productions having more than two symbols on the right hand side of the production. If X is a terminal say *a*, then replace this terminal by a corresponding non- terminal B_a and introduce the production

$$B_a \rightarrow a$$

The non-terminals on the right hand side of the production are retained. The resulting productions are added to P_1. The resulting context free grammar $G_1 = (V_1, T, P_1, S)$ where each production in P_1 is of the form

$$A \rightarrow A_1A_2\text{-----}A_n$$

or

$$A \rightarrow a$$

generates the same language as accepted by grammar G. So, $L(G) = L(G_1)$.

Step2: Restrict the number of variables on the right hand side of the production. Add all the productions of G_1 which are in CNF to P^1. Consider a production of the form

$$A \rightarrow A_1A_2\text{-------}A_n$$

where $n \geq 3$ (Note that if n = 2, the production is already in CNF and *n* can not be equal to 1. Because if n = 1, there is only one symbol and it is a terminal which again is in CNF). The A-production can be written as

$$\begin{aligned} A &\rightarrow A_1D_1 \\ D_1 &\rightarrow A_2D_2 \\ D_3 &\rightarrow A_3D_3 \\ &\vdots \\ D_{n-2} &\rightarrow A_{n-1}A_n \end{aligned}$$

These productions are added to P^1 and new variables are added to V^1. The grammar thus obtained is in CNF. The resulting grammar $G^1 = (V^1, T, P^1, S)$ generates the same language as accepted by G i.e. $L(G) = L(G^1)$.

Example 6.12: Consider the grammar

$$\begin{aligned} S &\rightarrow 0A \mid 1B \\ A &\rightarrow 0AA \mid 1S \mid 1 \\ B &\rightarrow 1BB \mid 0S \mid 0 \end{aligned}$$

Obtain the grammar in CNF.

All productions which are in CNF are added to P_1. The productions which are in standard form and added to P_1 are:

$$A \rightarrow 1$$
$$B \rightarrow 0 \qquad (6.8)$$

Consider the productions, which are not in CNF. Replace the terminal *a* on right hand side of the production by a non-terminal A and introduce the production A→ a. This step has to be carried out for each production which are not in CNF.

The table below shows the action taken indicating which terminal is replaced by the corresponding non-terminal and what is the new production introduced. The last column shows the resulting productions.

Given Productions	Action	Resulting productions
S→ 0A | 1B	Replace 0 by B_0 and introduce the production $B_0 \rightarrow 0$ Replace 1 by B_1 and introduce the production $B_1 \rightarrow 1$	$S \rightarrow B_0A \mid B_1B$ $B_0 \rightarrow 0$ $B_1 \rightarrow 1$
A→ 0AA/1S	Replace 0 by B_0 and introduce the production $B_0 \rightarrow 0$ Replace 1 by B_1 and introduce the production $B_1 \rightarrow 1$	$A \rightarrow B_0AA/B_1S$ $B_0 \rightarrow 0$ $B_1 \rightarrow 1$
B→1BB/0S	Replace 0 by B_0 and introduce the production $B_0 \rightarrow 0$ Replace 1 by B_1 and introduce the production $B_1 \rightarrow 1$	$B \rightarrow B_1BB/B_0S$ $B_1 \rightarrow 1$ $B_0 \rightarrow 0$

The grammar $G_1 = (V_1, T, P_1, S)$ can be obtained by combining the productions obtained from the last column in the table and the productions shown in (6.8).

$V_1 = \{S, A, B, B_0, B_1\}$
$T_1 = \{0, 1\}$
$P_1 = \{$

$S \rightarrow B_0A \mid B_1B$
$A \rightarrow B_0AA \mid B_1S \mid 1$
$B \rightarrow B_1BB \mid B_0S \mid 0$
$B_0 \rightarrow 0$
$B_1 \rightarrow 1$

$\}$

S is the start symbol

Step2: Restricting the number of variables on the right hand side of the production to 2. The productions obtained after step1 are:

$$\begin{aligned} S &\rightarrow B_0A \mid B_1B \\ A &\rightarrow B_0AA \mid B_1S \mid 1 \\ B &\rightarrow B_1BB \mid B_0S \mid 0 \\ B_0 &\rightarrow 0 \\ B_1 &\rightarrow 1 \end{aligned}$$

In the above productions, the productions which are in CNF are

$$\begin{aligned} S &\rightarrow B_0A \mid B_1B \\ A &\rightarrow B_1S \mid 1 \\ B &\rightarrow B_0S \mid 0 \\ B_0 &\rightarrow 0 \\ B_1 &\rightarrow 1 \end{aligned} \qquad (6.9)$$

and add these productions to P^1. The productions which are not in CNF are

$$\begin{aligned} A &\rightarrow B_0AA \\ B &\rightarrow B_1BB \end{aligned}$$

The following table shows how these productions are changed to CNF so that only two variables are present on the right hand side of the production.

Given Productions	Action	Resulting productions
$A \rightarrow B_0AA$	Replace AA on R.H.S with variable D_1 and introduce the production $D_1 \rightarrow AA$	$A \rightarrow B_0D_1$ $D_1 \rightarrow AA$
$B \rightarrow B_1BB$	Replace BB on R.H.S with variable D_2 and introduce the production $D_2 \rightarrow BB$	$B \rightarrow B_1D_2$ $D_2 \rightarrow BB$

(6.10)

The final grammar which is in CNF can be obtained by combining the productions in (6.9) and (6.10). The grammar $G^1 = (V^1, T, P^1, S)$ is in CNF where

$$\begin{aligned} V_1 &= \{S, A, B, B_0, B_1, D_1, D_2\} \\ T_1 &= \{0, 1\} \\ P_1 &= \{ \\ &\qquad S \rightarrow B_0A \mid B_1B \\ &\qquad A \rightarrow B_1S \mid 1 \mid B_0D_1 \end{aligned}$$

$$
\begin{aligned}
B &\rightarrow B_0S \mid 0 \mid B_1D_2 \\
B_0 &\rightarrow 0 \\
B_1 &\rightarrow 1 \\
D_1 &\rightarrow AA \\
D_2 &\rightarrow BB
\end{aligned}
$$

}

S is the start symbol

6.7 Greibach normal form (GNF):

In CNF there is restriction on the number of symbols on the right hand side of the production. Note that in CNF not more than two symbols on R.H.S of the production are permitted. If there is only symbol that symbol must be a terminal and if there are two symbols, those two symbols must be variables.

In GNF there is no restriction on the number of symbols on the right hand side, but there is restriction on the terminals and variables appear on the right hand side of the production.

Definition: Let G = (V, T, P, S) be a CFG. The CFG G is said to be in GNF if all the productions are of the form

$$A \rightarrow a\alpha$$

where $a \in T$ and $\alpha \in V^*$ i.e. the first symbol on the right hand side of the production must be a terminal and it can be followed by zero or more variables.

Theorem 6.6: Let G = (V, T, P, S) be a CFG generating the language L without ε. An equivalent grammar G1 generating the same language exists for which every production is of the form

$$A \rightarrow a\alpha$$

where A is a variable, *a* is a terminal and α is string of zero or more variables.

It means that any CFG can be converted into GNF notation. GNF is a very useful notation. If a grammar is in GNF, it can be easily converted into Pushdown Automaton which can accept only the context free languages. The Pushdown Automaton will be discussed in the next chapter. Now, let us see how to convert a given CFG to its equivalent GNF notation.

Procedure to obtain the grammar in GNF:

Step 1: Obtain the grammar in CNF.

Step 2: Rename the non-terminals to $A_1, A_2, A_3, \ldots\ldots$

Step 3: Using substitution method as discussed in section 6.1, obtain the productions to the form

$$A_i \rightarrow A_j\alpha \qquad \text{for } i < j$$

where $\alpha \in V^*$. Note if all the productions are in this manner, the number of steps will be reduced while converting.

Step 4: After substitution, if a grammar has left-recursion, we should eliminate left recursion as discussed in section 6.2.

Step 5: It may be necessary to apply step 3 and/or step 4 more than once to get the grammar in GNF.

Now, let us concentrate on how a grammar can be converted into GNF.

Example 6.13: Convert the following grammar

$$\begin{aligned} S &\rightarrow AB1 \mid 0 \\ A &\rightarrow 00A \mid B \\ B &\rightarrow 1A1 \end{aligned}$$

into GNF

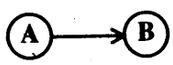
Note: A grammar should not have any unit productions and ε-productions. In case if the grammar has ε-productions and unit productions perform the following operations one after the other:

1. Eliminate all ε-productions.
2. Eliminate all unit productions
3. Obtain the grammar in CNF
4. Finally convert the grammar into GNF.

Since the grammar does not have any ε-productions, in the next let us eliminate the unit production

$$A \rightarrow B$$

The dependency graph for this can be



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

$$A \rightarrow 00A \mid B$$

the variable B can be replaced by the string 1A1 using the production

$$B \rightarrow 1A1$$

Now, the A-production can be written as

$$A \rightarrow 00A \mid 1A1$$

The resulting grammar obtained after eliminating unit productions is shown below:

$$\begin{aligned} S &\rightarrow AB1 \mid 0 \\ A &\rightarrow 00A \mid 1A1 \\ B &\rightarrow 1A1 \end{aligned}$$

Now, the grammar has to be converted into CNF. Now replace the terminals by non-terminals if they are not in CNF and the resulting grammar is

$$\begin{aligned} S &\rightarrow ABA_1 \mid 0 \\ A &\rightarrow A_0A_0A \mid A_1AA_1 \\ B &\rightarrow A_1AA_1 \\ A_1 &\rightarrow 1 \\ A_0 &\rightarrow 0 \end{aligned}$$

Now restrict the number of variables on the right hand side of the production to two and the resulting grammar in CNF notation is:

$$\begin{aligned} S &\rightarrow AD_1 \mid 0 \\ A &\rightarrow A_0D_2 \mid A_1D_3 \\ B &\rightarrow A_1D_3 \\ A_1 &\rightarrow 1 \\ A_0 &\rightarrow 0 \\ D_1 &\rightarrow BA_1 \\ D_2 &\rightarrow A_0A \\ D_3 &\rightarrow AA_1 \end{aligned}$$

Now let us rename all the variables as shown below:

Let $S = A_1$, $A = A_2$, $B = A_3$, $A_0 = A_4$, $A_1 = A_5$, $D_1 = A_6$, $D_2 = A_7$, $D_3 = A_8$. Now, the grammar can be re-written as

$$\begin{aligned} A_1 &\rightarrow A_2A_6 \mid 0 \\ A_2 &\rightarrow A_4A_7 \mid A_5A_8 \\ A_3 &\rightarrow A_5A_8 \\ A_5 &\rightarrow 1 \\ A_4 &\rightarrow 0 \\ A_6 &\rightarrow A_3A_5 \\ A_7 &\rightarrow A_4A_2 \\ A_8 &\rightarrow A_2A_5 \end{aligned}$$

In the above productions, note that A_4 and A_5-productions are in GNF.

Consider A_3 production: Substituting for A_5 in A_3-production we get

$$A_3 \rightarrow A_5A_8 = 1A_8$$

Now, A_3-production is in GNF.

Consider A_2 production: Since A_4 and A_5 productions are in GNF, substituting these productions in A_2-production we get

$$A_2 \rightarrow A_4A_7 \mid A_5A_8 = 0A_7 \mid 1A_8$$

Now, A_2-production is also in GNF.

Consider A_1 production: Since A_2 production is in GNF, substituting for A_2 in A_1-production we get

$$A_1 \rightarrow A_2A_6 \mid 0 = (0A_7 \mid 1A_8)A6 \mid 0 = 0A_7A_6 \mid 1A_8A_6 \mid 0$$

Now, A_1 production is also in GNF.

Consider A_6-productiion: Now, in A_6-production, replacing the first A_3 by A_3-production

$$A_6 \rightarrow A_3A_5 = (A_5A_8)A_5$$

we get the A_6 production

$$A_6 \rightarrow A_5A_8A_5$$

which is not in desired form. In the above production, replacing the first A_5 by A_5-production, we get A_6-production in GNF as shown below:

$A_6 \rightarrow A_5A_8A_5 = 1A_8A_5$

Consider A_7 production: Replacing the first A_4 in A_7-production we get

$$A_7 \rightarrow 0A_2$$

which is in GNF.

Consider A_8 production: Since A_2 production is in GNF, substituting for A_2 in A_8-production we get

$$A_8 \rightarrow A_2A_5 = (0A_7 \mid 1A_8)\,A_5 = 0A_7A_5 \mid 1A_8\,A_5$$

which is in GNF. Since all productions are in GNF, the whole grammar is in GNF. The final grammar which is in GNF is G = (V, T, P, S) where

V = $\{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\}$
T = $\{0, 1\}$
P = {
$A_1 \rightarrow 0A_7A_6 \mid 1A_8A_6 \mid 0$

$A_2 \rightarrow 0A_7 \mid 1A_8$
$A_3 \rightarrow 1A_8$
$A_4 \rightarrow 0$
$A_5 \rightarrow 1$
$A_6 \rightarrow 1A_8A_5$
$A_7 \rightarrow 0A_2$
$A_8 \rightarrow 0A_7A_5 \mid 1A_8A_5$

}

S is the start symbol

Note: In this problem there are only substitutions and after repeated substitutions, we get the grammar in GNF.

Example 6.14: Convert the following grammar

$A \rightarrow BC$
$B \rightarrow CA \mid b$
$C \rightarrow AB \mid a$

into GNF

Let $A = A_1$, $B = A_2$, $C = A_3$ and the resulting grammar is

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

First two productions are of the form

$$A_i \rightarrow A_j\alpha \qquad \text{for } i < j$$

So, we consider A_3-production.

Consider A_3-production: Substituting for A_1 in A_3-production we get

$$A_3 \rightarrow A_1A_2 \mid a \;=\; (A_2A_3)A_2 \mid a \;=\; A_2A_3A_2 \mid a$$

Again replacing the first A_2 in A_3-production we get,

$$A_3 \rightarrow A_2A_3A_2 \mid a \;=\; (A_3A_1 \mid b)\,A_3A_2 \mid a$$
$$=\; A_3A_1A_3A_2 \mid bA_3A_2 \mid a$$

we get the resulting A_3-production as

$$A_3 \rightarrow A_3A_1A_3A_2 \mid bA_3A_2 \mid a$$

which is having left recursion. After eliminating left recursion we get

$$A_3 \rightarrow bA_3A_2 \mid a \mid bA_3A_2Z \mid aZ$$
$$Z \rightarrow A_1A_3A_2 \mid A_1A_3A_2Z$$

Now, all A_3-productions are in GNF.

Consider A_2-productions: Since all A_3-productions are in GNF, substituting A_3-productions in A_2-production we get

$$A_2 \rightarrow (bA_3A_2 \mid a \mid bA_3A_2Z \mid aZ)A_1 \mid b$$
$$= bA_3A_2A_1 \mid aA_1 \mid bA_3A_2ZA_1 \mid aZA_1 \mid b$$

which is in GNF. Now, all A_2-productions are in GNF.

Consider A_1-productions: Since all A_2-productions are in GNF, substituting A_2-productions in A_1-production we get,

$$A_1 \rightarrow A_2A_3 = (bA_3A_2A_1 \mid aA_1 \mid bA_3A_2ZA_1 \mid aZA_1 \mid b)A_3$$
$$= bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3A_2ZA_1A_3 \mid aZA_1A_3 \mid bA_3$$

Now, A_1-productions are also in GNF.

Consider Z-productions: Since A_1-productions are in GNF, substituting A_1-production in Z-production we get Z-productions also in GNF as shown below:

$$Z \rightarrow A_1A_3A_2 \mid A_1A_3A_2Z$$
$$= (bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3A_2ZA_1A_3 \mid aZA_1A_3 \mid bA_3)A_3A_2$$
$$(bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3A_2ZA_1A_3 \mid aZA_1A_3 \mid bA_3)\, A_3A_2Z$$

which can be written as

$$Z \rightarrow bA_3A_2A_1A_3A_3A_2 \mid aA_1A_3A_3A_2 \mid bA_3A_2ZA_1A_3A_3A_2 \mid$$
$$aZA_1A_3\, A_3A_2 \mid bA_3A_3A_2$$
$$Z \rightarrow bA_3A_2A_1A_3A_3A_2Z \mid aA_1A_3A_3A_2Z \mid bA_3A_2ZA_1A_3A_3A_2Z \mid$$
$$aZA_1A_3\, A_3A_2Z \mid bA_3A_3A_2Z$$

Since all productions are in GNF, the resulting grammar is also in GNF. So, the final grammar obtained in GNF notation is

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

where

$V = \{A_1, A_2, A_3, Z\}$
$T = \{a, b\}$

P = {

$A_1 \rightarrow bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3A_2ZA_1A_3 \mid aZA_1A_3 \mid bA_3$
$A_2 \rightarrow bA_3A_2A_1 \mid aA_1 \mid bA_3A_2ZA_1 \mid aZA_1 \mid b$
$A_3 \rightarrow bA_3A_2 \mid a \mid bA_3A_2Z \mid aZ$
$Z \rightarrow bA_3A_2A_1A_3A_3A_2 \mid aA_1A_3A_3A_2 \mid bA_3A_2ZA_1A_3A_3A_2 \mid aZA_1A_3\ A_3A_2 \mid bA_3A_3A_2$
$Z \rightarrow bA_3A_2A_1A_3A_3A_2Z \mid aA_1A_3A_3A_2Z \mid bA_3A_2ZA_1A_3A_3A_2Z \mid aZA_1A_3\ A_3A_2Z \mid bA_3A_3A_2Z$

}

A_1 is the start symbol

Example 6.15: Convert the following grammar

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

into GNF

Let $S = A_1$ and $A = A_2$. After substitution, the resulting grammar obtained is shown below:

$A_1 \rightarrow A_2A_2 \mid 0$
$A_2 \rightarrow A_1A_1 \mid 1$

First production is of the form

$$A_i \rightarrow A_j\alpha \qquad \text{for } i < j$$

So, we consider A_2-production.

Consider A_2-production: Substituting for A_1 in A_2-production we get

$A_2 \rightarrow A_2A_2A_1 \mid 0A_1 \mid 1$

The above grammar is having left recursion. After eliminating left recursion we get

$A_2 \rightarrow 0A_1 \mid 1 \mid 0A_1Z \mid 1Z$
$Z \rightarrow A_2A_1 \mid A_2A_1Z$

Now, all A_2-productions are in GNF.

Consider A_1-productions: Since all A_2-productions are in GNF, substituting A_2-productions in A_1-production we get

$A_1 \rightarrow 0A_1A_2 \mid 1A_2 \mid 0A_1ZA_2 \mid 1ZA_2 \mid 0$

which is in GNF. Now, all A_1-productions are in GNF.

Consider Z-productions: Since A_2-productions are in GNF, substituting A_2-production in Z-production we get Z-productions also in GNF as shown below:

$Z \rightarrow 0A_1A_1 \mid 1A_1 \mid 0A_1ZA_1 \mid 1ZA_1$
$Z \rightarrow 0A_1A_1Z \mid 1A_1Z \mid 0A_1ZA_1Z \mid 1ZA_1Z$