Unit-III CONTEXT FREE GRAMMARS:

Every language is generated by some grammar. It produces strings of a language by means of replacing symbols. In general, a grammar is a set of production rules for generating strings in a language

Definition of a grammar:

A phrase structured grammar G (or simply grammar G) is a 4-tuple

$$G = (V_N, \sum, P, S) \text{ where}$$

(i) V_N is a finite nonempty set whose elements are called variables
(ii) $\sum$ is a finite nonempty set whose elements are called terminals
(iii) $S \in V_N$ is a special symbol called the start symbol
(iv) P is a finite set whose elements are of the form $\alpha \to \beta$ where
α and β are strings such that $\alpha, \beta \in (V_N \cup \sum)^*$. and α has atleast one symbol from V_N The elements of P are called productions or production rules
(v) $V_N \cap \sum = \phi$

Ex1:

$G = (V_N, \Sigma, P, S)$ is a grammar

where

$V_N = \{\langle\text{sentence}\rangle, \langle\text{noun}\rangle, \langle\text{verb}\rangle, \langle\text{adverb}\rangle\}$
$\Sigma = \{\text{Ram, Sam, ate, sang, well}\}$
$S = \langle\text{sentence}\rangle$

P consists of the following productions:

$\langle\text{sentence}\rangle \to \langle\text{noun}\rangle\ \langle\text{verb}\rangle$
$\langle\text{sentence}\rangle \to \langle\text{noun}\rangle\ \langle\text{verb}\rangle\ \langle\text{adverb}\rangle$
$\langle\text{noun}\rangle \to \text{Ram}$
$\langle\text{noun}\rangle \to \text{Sam}$
$\langle\text{verb}\rangle \to \text{ate}$
$\langle\text{verb}\rangle \to \text{sang}$
$\langle\text{adverb}\rangle \to \text{well}$

Ex 2: G = (V, T, P,S) where V={S} , T={a,b} and productions P are given as

$$\mathbf{S \to \varepsilon \mid a \mid b \mid aSa \mid bSb}$$

Def:

The language generated by a grammar G is denoted as L(G) and is defined as

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

i.e., L(G) is the set of all terminal strings derived from the start symbol S.

Chomsky classification of Grammars :

Chomsky classified the grammars into 4 types (type 0, type 1, type 2, type 3) based on types of productions.

Type 0 grammar(unrestricted grammar):

A grammar in which there are no restrictions on production rules is called a type 0 grammar.

Type 1 grammar(or context sensitive grammar):

A grammar in which all productions are of the form

$$\Phi A \Psi \rightarrow \Phi \alpha \Psi$$

where A is a variable (nonterminal), Φ , Ψ and α are strings of terminals and non terminals and $\alpha \neq \varepsilon$

The production $S \rightarrow \varepsilon$ is allowed in type 1 grammar but in that case S does not appear on the right hand side of any production.

Type 2 grammar(or context free grammar):

A grammar in which all productions are of the form

$$A \rightarrow \gamma$$

where A is a nonterminal, and γ is a string (potentially empty) of terminals and nonterminals

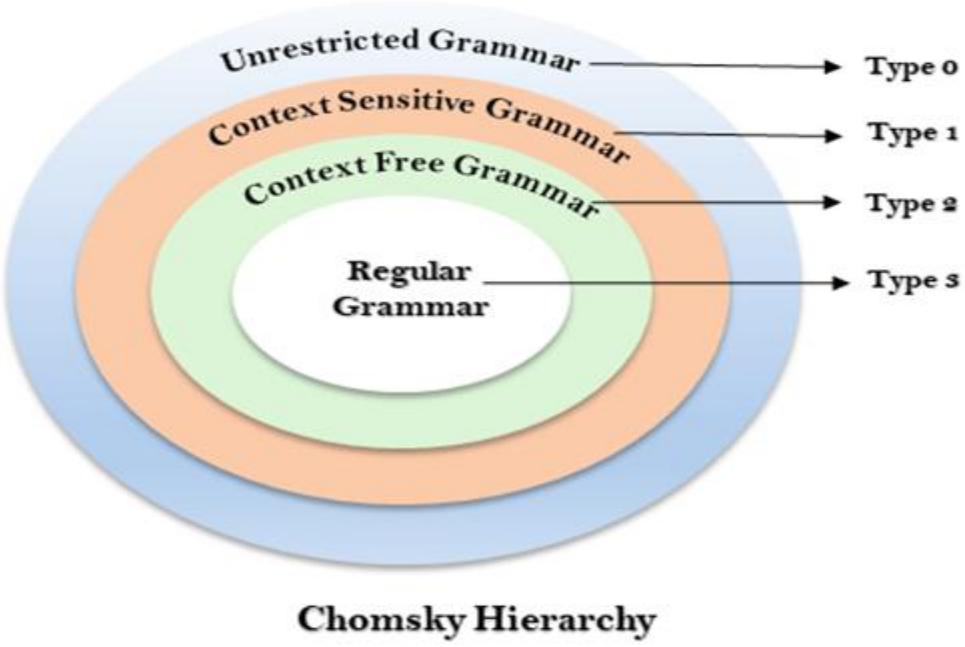
Type 3 grammar(or Regula grammar):

A grammar in which all productions are of the form

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

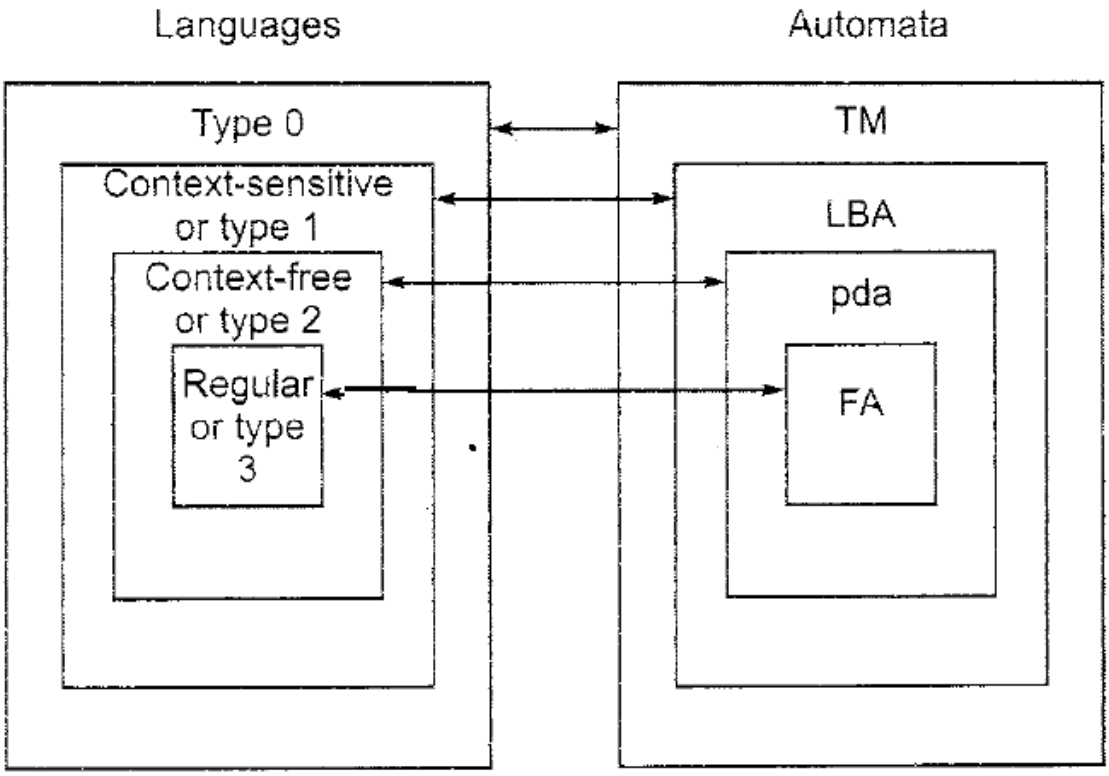
Where A is a nonterminal (or variable) and a is a terminal. The production $S \rightarrow \varepsilon$ is allowed in type 3 grammar but in that case S does not appear on the right hand side of any production

The four forms of grammars follows the chomsky hierarchy given below.



Chomsky Hierarchy

The languages generated by the grammars and corresponding automata that accepts the language is given below.



Context free languages:

The language generated by the context free grammar is called a context free language.

Ex:

Construct a CFG representing the set of palindromes over (0+1)*.

Sol:

The possible strings are

ε, 0, 1, 00, 11, 000, 111, etc

So the productions are

$$S \rightarrow \varepsilon$$
$$S \rightarrow 0$$
$$S \rightarrow 1$$
$$S \rightarrow 0S0$$
$$S \rightarrow 1S1$$

Thus the CFG is given by G = (V, T, P, S) where

$V = \{S\}$

$T = \{0, 1\}$

$P = \{ S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon\}$

$S = \{S\}$

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

The possible strings are

ε, ab, aabb, aaabbb, $a^4 b^4$, etc

So the productions are,

$$S \rightarrow \varepsilon$$
$$S \rightarrow aSb$$

Thus, the CFG, G = (V, T, P, S) is given by

$V = \{S\}$

$T = \{a, b\}$

$P = \{ S \rightarrow aSb \mid \varepsilon\}$

$S = \{S\}$

Let G = {{S, c}, {a,b}, P, S} where S → aCa, C → aCa|b. Find L(G).

Solution:

1. $S \rightarrow a\underline{C}a \rightarrow aba$
2. $S \rightarrow a\underline{C}a \rightarrow aa\underline{C}aa \rightarrow aabaa$
3. $S \rightarrow a\underline{C}aa \rightarrow aaa\underline{C}aaa \rightarrow aaabaaa$

.
.
.

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

DERIVATION AND LANGUAGES

Derivations

- Derivations are the set of strings that are derived from the start symbol, after applying the production rules, a finite number of times.

$$S \underset{G}{\overset{*}{\Rightarrow}} \omega \mid \omega \varepsilon T^*$$

Representation of derivations

- Derivations are represented in either of the two forms.
- They are,
  - Sentential form
  - Parse tree form

Types of derivations

There are two types of derivations, namely

- Leftmost derivations
- Rightmost derivations

Sentential form

- Sentential form is the derivation derived from the start symbol, by applying the rules/productions.
- Let the G = (V, T, P, S) be a CFG then

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

is in sentential form, where $\alpha \varepsilon (TUV)^*$

Ex:

Let G = (V, T, P, S) be given by G = ({S, A, B}, {0, 1}, P, {S}).

P: $S \rightarrow A1B$

$A \rightarrow 0A \mid \varepsilon$

$B \rightarrow 0B \mid 1B \mid \varepsilon$

Determine whether the string 00101 is an element of L(G) or not

$S \Rightarrow \underline{A}1B$ [Using $S \rightarrow A1B$]

$\Rightarrow 0\underline{A}1B$ [Using $A \rightarrow 0A$]

$\Rightarrow 00\underline{A}1B$ [Using $A \rightarrow 0A$]

$\Rightarrow 001\underline{B}$ [Using $A \rightarrow \varepsilon$]

$\Rightarrow 0010\underline{B}$ [Using $B \rightarrow 0B$]

$\Rightarrow 00101\underline{B}$ [Using $B \rightarrow 1B$]

$\Rightarrow 00101$ [Using $B \rightarrow \varepsilon$]

Thus the string 00101 $\in$ L(G).

Left most derivation:

A derivation

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

is called a leftmost derivation if we apply a production only to the leftmost variable at each step

Rightmost derivation:

A derivation

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

is called a Rightmost derivation if we apply a production only to the Rightmost variable at each step

Ex1:

Let G be the grammar. P = {S → aB | bA, A → a | aS | bAA, B → b | bs | aBB}. For the string aabbbbaa, find LMD and RMD.

Leftmost derivation

$$S \underset{\ell m}{\Rightarrow} a\underline{B}$$

$$\underset{\ell m}{\Rightarrow} aa\underline{B}B$$

$$\underset{\ell m}{\Rightarrow} aab\underline{B}$$

$$\underset{\ell m}{\Rightarrow} aabb\underline{S}$$

$$\underset{\ell m}{\Rightarrow} aabbb\underline{A}$$

$$\underset{\ell m}{\Rightarrow} aabbbbAA$$

$$\underset{\ell m}{\Rightarrow} aabbbbaA$$

$$\underset{\ell m}{\Rightarrow} aabbbbaa$$

Rightmost derivation

$$S \underset{rm}{\Rightarrow} aB$$

$$\underset{rm}{\Rightarrow} aaB\underline{B}$$

$$\underset{rm}{\Rightarrow} aaBb\underline{S}$$

$$\underset{rm}{\Rightarrow} aaBbb\underline{A}$$

$$\underset{rm}{\Rightarrow} aaBbbbA\underline{A}$$

$$\underset{rm}{\Rightarrow} aaBbbb\underline{A}a$$

$$\underset{rm}{\Rightarrow} aa\underline{B}bbbaa$$

$$\underset{rm}{\Rightarrow} aabbbbaa$$

Ex2:

For the CFG,G=({S},{a,b},P,S) where P is given as

$S \rightarrow \varepsilon$

$S \rightarrow SS$

$S \rightarrow aSb$

$S \rightarrow bSa$

Find the leftmost and rightmost derivation for the string 'abab'

Sol:

Leftmost derivation:

$S \underset{\ell m}{\Rightarrow} a\underline{S}b$ [Since $S \rightarrow aSb$]

$\underset{\ell m}{\Rightarrow} ab\underline{S}ab$ [Since $S \rightarrow bSa$]

$\underset{\ell m}{\Rightarrow} abab$ [Since $S \rightarrow \varepsilon$]

Rightmost derivation:

$S \underset{rm}{\Rightarrow} S\underline{S}$ [Since $S \rightarrow SS$]

$\underset{rm}{\Rightarrow} Sa\underline{S}b$ [Since $S \rightarrow aSb$]

$\underset{rm}{\Rightarrow} a\underline{S}ab$ [Since $S \rightarrow \varepsilon$]

$\underset{rm}{\Rightarrow} a\underline{S}bab$ [Since $S \rightarrow aSb$]

$\underset{rm}{\Rightarrow} abab$ [Since $S \rightarrow \varepsilon$]

Parse tree (or) derivation trees:

The derivations in a CFG can be represented using trees. Such trees representing the derivations are called parse trees or derivation trees.

Definition A derivation tree (also called a parse tree) for a CFG $G = (V_N, \Sigma, P, S)$ is a tree satisfying the following conditions:

(i) Every vertex has a label which is a variable or terminal or Λ.
(ii) The root has label S.
(iii) The label of an internal vertex is a variable.
(iv) If the vertices $n_1, n_2, \ldots, n_k$ written with labels $X_1, X_2, \ldots, X_k$ are the sons of vertex n with label A, then $A \rightarrow X_1X_2 \ldots X_k$ is a production in P.
(v) A vertex n is a leaf if its label is $a \in \Sigma$ or Λ;

Example

For the grammar S → 0S1|01, generate the derivation of the string '000111'.

Solution:

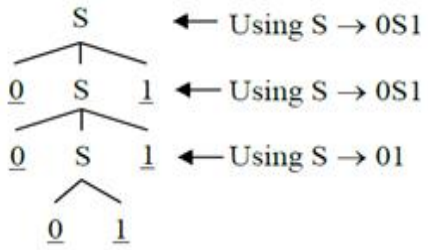
The derivation is given by

$S \Rightarrow 0S1$ [Using $S \rightarrow 0S1$]

$S \Rightarrow 00S11$ [Using $S \rightarrow 0S1$]

$S \Rightarrow 000111$ [Using $S \rightarrow 01$]

The parse tree is given by



Yield of a parse tree

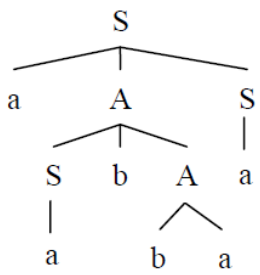
- The terminals present at the leaf nodes are concatenated from left to right generate a string, derived from the grammar.
- The generated string is called the yield of a parse tree.

Example

- In the above mentioned parse tree, the terminal symbols in the leaf nodes, taken from left to right generate the string, „000111".
- Thus 000111 is the yield of the parse tree.

Construct a parse tree for S ⇒ aabbaa, given the productions: S → aAS|a; A → SbA|SS|ba.

Solution:



[Using S → aAS]

[Using A → SbA and S → a]

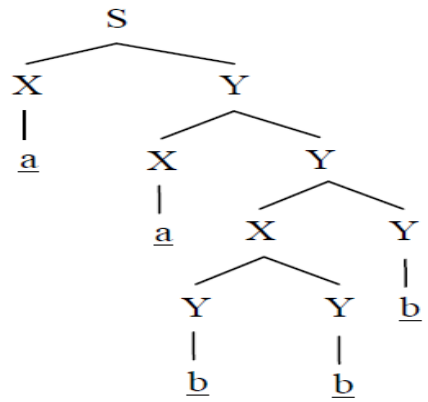
[Using S → a and A → ba]

Find whether S = aabbb is in L = L(G) where G is given by S → XY; X → YY | a; Y → XY | b.

Sentential form

$S \Rightarrow \underline{X}Y$	[Using S → XY]
$\Rightarrow a\underline{Y}$	[Using X → a]
$\Rightarrow a\underline{X}Y$	[Using Y → XY]
$\Rightarrow aa\underline{Y}$	[Using X → a]
$\Rightarrow aa\underline{X}Y$	[Using Y → XY]
$\Rightarrow aa\underline{Y}YY$	[Using X → YY]
$\Rightarrow aab\underline{Y}Y$	[Using Y → b]
$\Rightarrow aabb\underline{Y}$	[Using Y → b]
$\Rightarrow aabbb$	[Using Y → b]

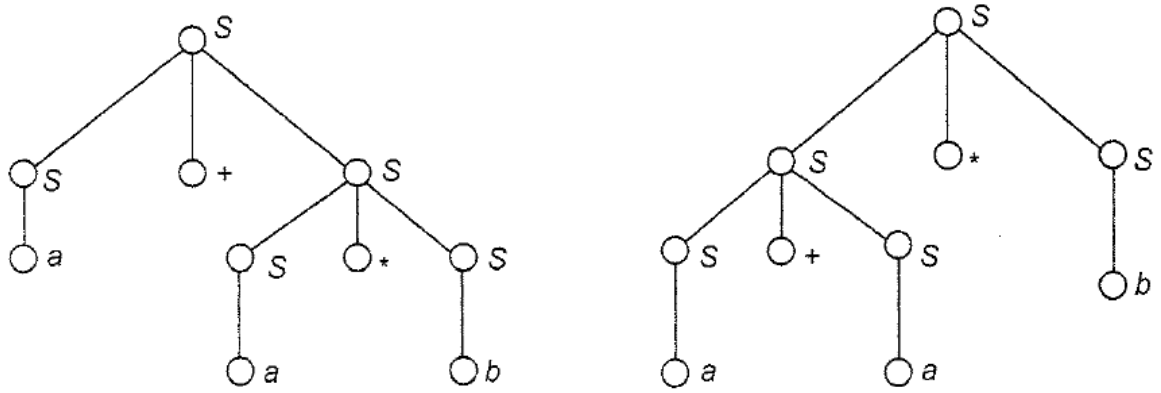
Parse tree



Ambiguous Grammars:

Definition A terminal string $w \in L(G)$ is ambiguous if there exist two or more derivation trees for w (or there exist two or more leftmost derivations of w).

Consider, for example, $G = (\{S\}, \{a, b, +, *\}, P, S)$, where P consists of $S \rightarrow S + S \mid S * S \mid a \mid b$. We have two derivation trees for $a + a * b$ given in Fig.



Therefore, $a + a * b$ is ambiguous.

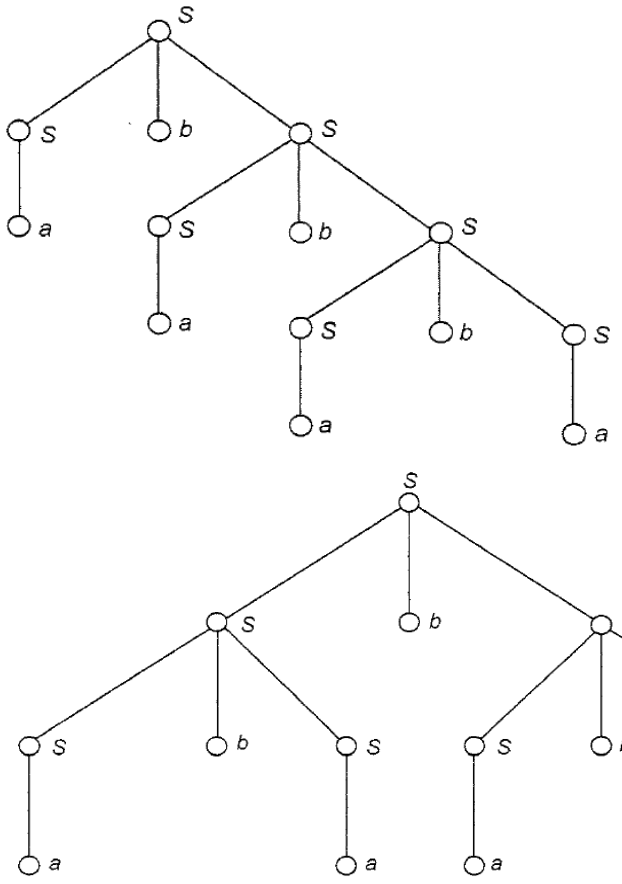
Definition A context-free grammar G is ambiguous if there exists some $w \in L(G)$, which is ambiguous.

Ex1:

If G is the grammar $S \rightarrow SbS \mid a$, show that G is ambiguous.

To prove that G is ambiguous, we have to find a $w \in L(G)$, which is ambiguous. Consider $w = abababa \in L(G)$.

Now we can get two derivation trees for w as shown below



Therefore the grammar G is ambiguous grammar.

Simplification of context free grammars :

Reduced grammar:

A reduced grammar 'G' has the following properties

1. Each terminals and nonterminals that appear in G must appear at least once in the derivation of some string accepted by L(G)

2. It does not contain the null productions i.e., productions of the form
   A $\longrightarrow$ ε where A is a nonterminal

3. It does not contain the unit productions i.e., productions of the form
   A $\longrightarrow$ B where A and B are nonterminals

Any grammar that satisfies the above conditions then it is said to be in reduced form

To get the reduced grammar, we have to do the following

1. Eliminate null productions

2. Eliminate the unit productions

3 .Eliminate useless symbols

Elimination of null productions:

Nullable variable:

A variable is a nullable variable if

$$A \stackrel{*}{\Rightarrow} \epsilon.$$

Finding nullable Variables:

Let G=(V,T,P,S) be a CFG. We can find all the nullable symbols of G by the following algorithm.

Basis:

If $A \rightarrow \varepsilon$ is a production of G, then A is nullable

Induction:

If there is a production $B \rightarrow C_1C_2.....C_k$, where each C_i is nullable, then B is also nullable.

Note:

Now we give the construction of a grammar without ϵ-productions. Let $G = (V, T, P, S)$ be a CFG. Determine all the nullable symbols of G. We construct a new grammar $G_1 = (V, T, P_1, S)$, whose set of productions P_1 is determined as follows.

For each production $A \to X_1 X_2 \cdots X_k$ of P, where $k \geq 1$, suppose that m of the k X_i's are nullable symbols. The new grammar G_1 will have 2^m versions of this production, where the nullable X_i's, in all possible combinations are present or absent. There is one exception: if $m = k$, i.e., all symbols are nullable, then we do not include the case where all X_i's are absent. Also, note that if a production of the form $A \to \epsilon$ is in P, we do not place this production in P_1.

Theorem : If the grammar G_1 is constructed from G by the above construction for eliminating ϵ-productions, then $L(G_1) = L(G) - \{\epsilon\}$.

Ex:

Eliminate the null productions from the grammar

$$S \to AB,$$
$$A \to aAA \mid \varepsilon$$
$$B \to bBB \mid \varepsilon$$

Sol:

Since $A \to \varepsilon, B \to \varepsilon$ *are productions of the grammar*, the variables A and B are nullable. Since A and B are nullable and S ⟶ AB is a production, S is also nullable. Therefore the nullable variables are {S,A,B}

After removing null productions, the grammar is given as

$$S \to AB \mid A \mid B$$
$$A \to aAA \mid aA \mid a$$
$$B \to bBB \mid bB \mid b$$

Eliminate the null productions from the grammar

$$S \to ABA,$$
$$A \to 0A \mid \varepsilon$$
$$B \to 1B \mid \varepsilon$$

Sol:

Since $A \to \varepsilon, B \to \varepsilon$ *are productions of the grammar*, the variables A and B are nullable. Since A and B are nullable and S ⟶ ABA is a production, S is also nullable. Therefore the nullable variables are {S,A,B}

After removing null productions, the grammar is given as

$$S \to ABA \mid BA \mid AA \mid AB \mid A \mid B$$
$$A \to 0A \mid 0$$
$$B \to 1B \mid 1$$

Eliminating unit productions:

A *unit production* is a production of the form $A \to B$, where both A and B are variables.

Unit pair:

A pair (A , B) is called as a unit pair if $A \overset{*}{\Rightarrow} B$ by using only unit productions.

Finding unit pairs:

BASIS: (A, A) is a unit pair for any variable A. That is, $A \overset{*}{\Rightarrow} A$ by zero steps.

INDUCTION: Suppose we have determined that (A, B) is a unit pair, and $B \to C$ is a production, where C is a variable. Then (A, C) is a unit pair.

Note:

To eliminate unit productions, we proceed as follows. Given a CFG $G = (V, T, P, S)$, construct CFG $G_1 = (V, T, P_1, S)$:

1. Find all the unit pairs of G.
2. For each unit pair (A, B), add to P_1 all the productions $A \to \alpha$, where $B \to \alpha$ is a nonunit production in P. Note that $A = B$ is possible; in that way, P_1 contains all the nonunit productions in P.

Theorem : If grammar G_1 is constructed from grammar G by the algorithm described above for eliminating unit productions, then $L(G_1) = L(G)$.

Ex: Eliminate the unit productions from the following grammar

$I \rightarrow a \mid b \mid Ia \mid Ib$
$F \rightarrow I \mid (S)$
$M \rightarrow F \mid M * F$
$S \rightarrow M \mid S + M$

Sol : Here the unit productions are $F \rightarrow I$, $M \rightarrow F$ and $S \rightarrow M$

Finding all the unit pairs and the production rules to be added are given below.

Unit pair	Production rules added
(I, I)	$I \rightarrow a \mid b \mid Ia \mid Ib$
(F, F)	$F \rightarrow (S)$
(M, M)	$M \rightarrow M * F$
(S, S)	$S \rightarrow S + M$
(F, I)	$F \rightarrow a \mid b \mid Ia \mid Ib$
(M, I)	$M \rightarrow a \mid b \mid Ia \mid Ib$
(M, F)	$M \rightarrow (S)$
(S, I)	$S \rightarrow a \mid b \mid Ia \mid Ib$
(S, F)	$S \rightarrow (S)$
(S, M)	$S \rightarrow M * F$

After eliminating unit productions, the grammar is

$I \rightarrow a \mid b \mid Ia \mid Ib$
$F \rightarrow a \mid b \mid Ia \mid Ib \mid (S)$
$M \rightarrow a \mid b \mid Ia \mid Ib \mid (S) \mid M * F$
$S \rightarrow a \mid b \mid Ia \mid Ib \mid (S) \mid M * F \mid S + M$

Ex2 : Eliminate the unit productions from the following grammar

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid a$$

Sol:
Here the unit productions are $E \rightarrow T$ and $T \rightarrow F$

By using unit pair algorithm, the unit pairs are (E,E), (E,T), (E,F), (T,T), (T,F), (F,F).

Now the production rules to be added are given as follows.

Unit pair	production rules to be added
(E,E)	$E \rightarrow E + T$
(T,T)	$T \rightarrow T * F$
(F,F)	$F \rightarrow (E) \mid a$
(E,T)	$E \rightarrow T * F$
(E,F)	$E \rightarrow (E) \mid a$
(T,F)	$T \rightarrow (E) \mid a$

The equivalent grammar with no unit productions is:

$$E \rightarrow E + T \mid T * F \mid (E) \mid a$$
$$T \rightarrow T * F \mid (E) \mid a$$
$$F \rightarrow (E) \mid a$$

Elimination of useless symbols:

A symbol is *useful* if it appears in some derivation of some terminal string from the start symbol.

(or)

We say a symbol X is *useful* for a grammar $G = (V, T, P, S)$ if there is some derivation of the form $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$, where w is in T^*.

Here X may be a terminal or nonterminal.

Note:
1. If the symbol X is not useful, it is a useless symbol.

2. A symbol X is a useful symbol if it is both generating symbol and reachable symbol.

3. X is generating if $X \overset{*}{\Rightarrow} w$ for some terminal string w. Every terminal symbol is a generating symbol

4. X is reachable if there is a derivation $S \overset{*}{\Rightarrow} \alpha X \beta$ for some α and β

Computing the generating symbols :

Let G =(V,T,P,S) be a grammar. To compute the generating symbols of G, we perform the following induction

Basis:

Every symbol of T is obviously generating; it generates itself

Induction:

Suppose there is a production $A \rightarrow \alpha$, and if every symbol of α is already known to be generating, then A is generating. (This rule includes the case where $\alpha = \varepsilon$)

Computing the reachable symbols:

Let G =(V,T,P,S) be a grammar. To compute the reachable symbols of G, we perform the following induction

Basis:

S is surely reachable

Induction:

Suppose we have discovered that some variable A is reachable, then for all productions with A in the head, all the symbols of the bodies of those productions are also reachable

Note:

Theorem : Let $G = (V, T, P, S)$ be a CFG, and assume that $L(G) \neq \emptyset$; i.e., G generates at least one string. Let $G_1 = (V_1, T_1, P_1, S)$ be the grammar we obtain by the following steps:

1. First eliminate nongenerating symbols and all productions involving one or more of those symbols. Let $G_2 = (V_2, T_2, P_2, S)$ be this new grammar. Note that S must be generating, since we assume $L(G)$ has at least one string, so S has not been eliminated.

2. Second, eliminate all symbols that are not reachable in the grammar G_2.

Then G_1 has no useless symbols, and $L(G_1) = L(G)$.

Note:
The order is important. Looking first for non-reachable symbols and then for non-generating symbols can still leave some useless symbols.

Ex:

Find the useless symbols from the grammar

$$S \rightarrow AB\ ,\ A \rightarrow a,\ B \rightarrow b,\ B \rightarrow C,\ E \rightarrow c$$

Sol: A symbol is a useful symbol if it is both generating and reachable symbol.

Finding the generating symbols:

The terminal symbols a, b and c are generating symbols.

Also the variables A is generating since there is production $A \rightarrow a$

The symbol B is also generating since there is a production $B \rightarrow b$

The symbol E is also generating since there is a production $E \rightarrow c$

Since A and B are generating and there is a production $S \rightarrow AB$, the symbol S is generating symbol

The generating symbols are {S,A,B,a,b,c}

The symbol C is not generating so it is a useless symbol

Eliminating the symbol C and its associated productions, the grammar is given as

$$S \rightarrow AB\ ,\ A \rightarrow a,\ B \rightarrow b,\ E \rightarrow c$$

Finding the reachable symbols:

Clearly S is reachable symbol.

Since there is a production $S \rightarrow AB$ and S is reachable, the symbols A and B are reachable.
Since A is reachable and $A \rightarrow a$ is a production, the symbol a is reachable
Since B is reachable and $B \rightarrow b$ is a production, the symbol b is reachable

So the set of reachable symbols are {S,A,B,a,b}

Therefore, the symbols that are not reachable are E and c. so the symbols E and c are useless.
Eliminating the useless symbols and its corresponding productions, the reduced grammar

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

Ex2:

Find a reduced grammar equivalent to the grammar G whose productions are

$$S \rightarrow AB \mid CA, \qquad B \rightarrow BC \mid AB, \qquad A \rightarrow a, \qquad C \rightarrow aB \mid b$$

Ans:

To get the reduced grammar, we have to do the following

1. Eliminate null productions

2. Eliminate the unit productions

3 .Eliminate useless symbols

The given grammar does not contain unit productions and null productions.

Eliminating useless symbols:

A symbol is a useful symbol if it is both generating and reachable symbol.

Finding the generating symbols:

The terminal symbols a, b are generating symbols.

Also the variables A is generating since there is production A → a

The symbol C is also generating since there is a production C → b

Since A and C are generating and there is a production S → CA, the symbol S is generating symbol

The generating symbols are {S,A,C,a,b}

The symbol Bis not generating so it is a useless symbol

Eliminating the symbol B and its associated productions, the grammar is given as

S → CA , A → a, C → b

Finding the reachable symbols:

Clearly S is reachable symbol.

Since there is a production S → CA and S is reachable, the symbols A and C are reachable. Since A is reachable and A → a is a production, the symbol a is reachable

Since C is reachable and C → b is a production, the symbol b is reachable

So the set of reachable symbols are {S,A,C,a,b}

The set of useful symbols are {S,A,C,a,b}. Therefore, the reduced grammar is given as

$$G' = (\{ S,A,C\}, \{a,b\}, \{ S \rightarrow CA , A \rightarrow a, C \rightarrow b\}, S)$$

Normal Forms:

When the productions in G satisfy certain restrictions, then G is said to be in a 'normal form'. Some of the normal forms are

a) Chomsky Normal Form(CNF)
b) Greibach Normal Form(GNF)

Chomsky Normal Form(CNF):

A CFG is said to be in *Chomsky Normal Form* if every production is of form

$A \rightarrow BC$ (right side is two variables).
(or)
$A \rightarrow a$ (right side is a single terminal).

where a is a terminal, and A, B, and C are variables.

Here $S \rightarrow \varepsilon$ is in G if $\varepsilon \in L(G)$. When ε is in *L(G)*, we assume that S does not appear on the R.H.S. of any production.

Procedure to convert CFG to CNF:

Step1:
Convert the given CFG to reduced form by eliminating null productions, unit productions and useless symbols.

Step 2:
For each terminal a, introduce a new variable, C_a (say), add a rule $C_a \rightarrow a$, and for each occurrence of a in a string of length 2 or more on the right hand side of a rule, replace a by C_a.

Step3:
For each rule with 3 or more variables on the right hand side, we replace it with a new collection of rules obeying required criteria as follows. Consider production of the form $A \rightarrow A_1A_2 \ldots A_m$ for some $m \geq 3$. By introducing new variables $C_1, C_2, \ldots\ldots\ldots\ldots, C_{m-2}$, the production is replaced by new productions

$A \rightarrow A_1C_1$,
$C_1 \rightarrow A_2C_2$,
……,
$C_{m-2} \rightarrow A_{m-1}A_m$

Ex:
Convert the given grammar to CNF-

S → 1A / 0B
A → 1AA / 0S / 0
B → 0BB / 1S / 1

Sol:
The given grammar is in reduced form.
The **productions already in chomsky normal form** are-
A → 0 ……..(1)
B → 1 ……..(2)
These productions will remain as they are.

The **productions not in chomsky normal form** are-
S → 1A / 0B ……..(3)
A → 1AA / 0S ……..(4)
B → 0BB / 1S ……..(5)
We will convert these productions in chomsky normal form.
Replace the terminal symbols 0 and 1 by new variables C and D.
This is done by introducing the following two new productions in the grammar-

C → 0 ……..(6)
D → 1 ……..(7)

Now, the productions (3), (4) and (5) modifies to-
S → DA / CB ……..(8)
A → DAA / CS ……..(9)
B → CBB / DS ……..(10)

Out of (8), (9) and (10), the productions already in Chomsky Normal Form are-
S → DA / CB ……..(11)
A → CS ……..(12)
B → DS ……..(13)
These productions will remain as they are.

The productions not in chomsky normal form are-
A → DAA ……..(14)
B → CBB ……..(15)

We will convert these productions in Chomsky Normal Form.Replace AA and BB by new variables E and F respectively.

This is done by introducing the following two new productions in the grammar-
E → AA ……..(16)
F → BB ……..(17)

Now, the productions (14) and (15) modifies to-

A → DE ……..(18)
B → CF ……..(19)

From (1), (2), (6), (7), (11), (12), (13), (16), (17), (18) and (19), the Chomsky normal form of the given grammar is-
S → DA / CB
A → CS / DE / 0
B → DS / CF / 1
C → 0
D → 1
E → AA
F → BB

Greibach Normal Form(GNF):

A grammar is in GNF if every production is of the form

$$A \rightarrow a\alpha, \text{ where } \alpha \in V^* \text{ and } a \in \Sigma .$$

Here $S \rightarrow \varepsilon$ is in G if $\varepsilon \in L(G)$. When ε is in $L(G)$, we assume that S does not appear on the R.H.S. of any production.

Note:

Lemma1: Let G be a CFG, and $A \rightarrow B\lambda$ be a production in P. Assume that P also has the following productions:

$B \rightarrow \beta_1|\beta_2|....|\beta_s$

We can define $P_1=(P - \{A \rightarrow B\lambda\}) \cup \{A \rightarrow \beta_i\lambda | 1 \leq i \leq s\}$

Then G1 having P1 is also a CFG equivalent to G.

Lemma2: Let G be a CFG contains productions of the form

$A \rightarrow A\alpha_1| A\alpha_2|....| A\alpha_r|\beta_1|\beta_2|....|\beta_s$ (βi's do not start with A)

Let Z be a new variable in G1 where P1 is defined as follows:

$A \rightarrow \beta_1|\beta_2|....|\beta_s$

$A \rightarrow \beta_1Z|\beta_2Z|....|\beta_sZ.$

$Z \rightarrow \alpha_1|\alpha_2|....|\alpha_r$

$Z \rightarrow \alpha_1Z|\alpha_2Z|....|\alpha_rZ$

Then G1 is a CFG and equivalent to G.

Procedure to convert the grammar to GNF:

Step1: Check whether the given grammar is in CNF. If not in CNF, make it in CNF. Rename the variables as $A_1, A_2, \ldots A_n$. With $S=A_1$.

Step2: A_i productions should be of the form $A_i \rightarrow A_j\lambda$ such that $i < j$. If there are productions of the form $A_i \rightarrow A_i\lambda$, apply Lemma2 to get rid of such productions. Otherwise apply Lemma1.

Step3: Modify Z_i productions. Apply Lemma1 to eliminate productions of the form $Z_i \rightarrow A_k\lambda$.

Ex:
Construct a grammar in GNF equivalent to the grammar:
$S \rightarrow AA \mid a$, $A \rightarrow SS \mid b$

Sol:
The given grammar is in CNF. Renaming the variables S and A as A_1 and A_2 respectively, the productions are

$A_1 \rightarrow A_2A_2 \mid a$
$A_2 \rightarrow A_1A_1 \mid b$

A_1 productions are in the required form. $A_2 \rightarrow b$ is also in the required form.

Applying known Lemma to $A_2 \rightarrow A_1A_1$, the resulting productions are
$A_2 \rightarrow A_2A_2A_1 \quad A_2 \rightarrow aA_1$

Now the A2- productions are
$A_2 \rightarrow A_2A_2A_1 \mid aA_1 \mid b$

By introducing the new variable Z_2 and Applying known lemma to the production $A_2 \rightarrow A_2A_2A_1$, the resulting productions are

$A_2 \rightarrow aA_1, \quad A_2 \rightarrow b$

$A_2 \rightarrow aA_1Z_2, A2 \rightarrow bZ_2$

$Z_2 \rightarrow A_2A_1, \quad Z_2 \rightarrow A_2A_1Z_2$

Now the A_2- productions are $A_2 \rightarrow aA_1 \mid b \mid aA_1Z_2 \mid bZ_2$

Eliminating the production $A_1 \rightarrow A_2A_2$ using known Lemma, the resulting productions are

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

Now the A_1 – productions are

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

Now the productions of the grammar are

$A_1 \rightarrow a \mid aA_1A_2 \mid bA_2 \mid aA_1Z_2A_2 \mid bZ_2A_2$
$A_2 \rightarrow aA_1 \mid b \mid aA_1Z_2 \mid bZ_2$
$Z_2 \rightarrow A_2A_1, \quad Z_2 \rightarrow A_2A_1Z_2$

Now the Z_2 -productions $Z_2 \rightarrow A_2A_1, \quad Z_2 \rightarrow A_2A_1Z_2$ are to be modified

Applying known lemma, we get the productions

$Z_2 \rightarrow aA_1A_1 \mid bA_1 \mid aA_1Z_2A_1 \mid bZ_2A_1$

$Z_2 \rightarrow aA_1A_1Z_2 \mid bA_1Z_2 \mid aA_1Z_2A_1Z_2 \mid bZ_2A_1Z_2$

Hence equivalent grammar is $G_1 = (\{A_1, A_2, Z_2\}, \{a, b\}, p_1, A_1)$
Where p_1 consists of productions

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

Pumping Lemma for Context Free Languages:

Let L be a CFL. Then there exists a constant n such that if $z \in L$ with $|z| \geq n$, then we can write $z = uvwxy$, satisfying the following conditions

1. $|vwx| \leq n$.
2. $vx \neq \varepsilon$
3. For all $i \geq 0$, we have $uv^iwx^iy \in L$.

This lemma is used to show that certain languages are not Context free languages. The steps needed to show that certain language is not context free are given below.

Step 1: Assume that L is context free language. Let n be the natural number obtained by pumping lemma.

Step 2: choose a string z such that $|z| \geq n$. Use pumping lemma to write $z=uvwxy$

Step 3: Find a suitable integer k such that $uv^kwx^ky \notin L$. This contradicts our assumption that L is context free. Hence L is not a context free language.

Ex:
Show that the language $L = \{ a^p \mid p \text{ is a prime} \}$ is not context Free

Sol: suppose that L is a context free language. Let n be the natural number obtained by using the pumping lemma.

Let $p > n$ be a prime number. Then the string $z = a^p \in L$. By pumping lemma we can write $z = uvwxy$

Also for any $k \geq 0$, we have $uv^kwx^ky \in L$.

For k=0, $uv^0wx^0y = uwy \in L$. so $|uwy|$ is a prime number, say q. let $|vx| = r$.

Now $|uv^qwx^qy| = |uwy| + q|vx| = q + qr = q(1+r)$

As $q(1+r)$ is not a prime, $uv^kwx^ky \notin L$ which is a contradiction which arises because of our assumption that L is context free. Hence L is not a context free language.

Ex: Show that the language $L = \{ 0^n1^n2^n \mid n \geq 1\}$ is not context Free

Sol:

suppose that L is a context free language. Let n be the natural number obtained by using the pumping lemma.

Let $z=0^n1^n2^n$. As $|z| \geq n$, $z \in L$. By pumping lemma we can write $z = uvwxy$ with $|vwx| \leq n$ and $vx \neq \varepsilon$

As $|vwx| \leq n$, Now vwx cannot contain both 0's and 2's The following cases are possible.

Case 1: vwx has no 2's. But then uwy cannot be in L since at least one of v or x is nonempty.
Case 2: vwx has no 0's. Again similarly, uwy cannot be in L.

In both cases, we have a contradiction which arises because of our assumption that L is context free. Hence L is not a context free language.

Closure Properties of context free languages:

The closure properties of context free languages are given below.

1. The union of two Context Free Languages is a Context Free Language i.e., if L_1 and L_2 are two context free languages then $L_1 \cup L_2$ is a context Free Language.

2. The concatenation of two Context Free Languages is a Context Free Language i.e., if L_1 and L_2 are two context free languages then L_1L_2 is a context Free Language.

3. The reversal of a Context Free Language is a Context Free Language i.e., if L is a CFL then L^R is also a CFL

4. The Closure(Kleene) of a Context Free Language is a Context Free Language i.e., if L is a CFL then L* is also a CFL

The Context Free Languages are not closed under Intersection , Complementation and difference i.e.,

5. The Intersection of two Context Free Languages need not be a Context Free Language

6. The Complement of a Context Free Language need not be a Context Free Language

7. The difference of two Context Free Languages need not be a Context Free Language

Note:

1. If L is a CFL and R is a regular language, then $L \cap R$ is a CFL

2. If L is a CFL and R is a regular language, then L - R is a CFL

Applications of Context Free Grammars:

1. The CFG is an essential concept for the implementation of compilers and other programming language processors. Tools such as YACC take a CFG as input and produce a parser

2. An essential part of XML is the Document Type Definition(DTD) which is essentially a CFG that describes the allowable tags and the ways in which these tags may be nested.