Unit - II

Content: Free Grammars and Languages

Definition of a grammar:

A phrase-structure grammar (or) simply a grammar is 

\((V_N, \Sigma, P, S)\), where

i) \(V_N\) is a finite nonempty set whose elements are called

variables or non terminals.

ii) \(\Sigma\) is a finite non empty set whose elements are called

terminals.

iii) \(V_N \cup \Sigma = \Phi\)

iv) \(S\) is a special variable (ie an element of \(V_N\)) called

start symbol.

v) \(P\) is a finite set whose elements are \(\alpha \rightarrow \beta\), where \(\alpha\)

and \(\beta\) are string on \(V_N \cup \Sigma\) \(\alpha\) has atleast one

symbol that is not in \(V_N\). Elements of \(P\) are called

productions or production rules or rewriting rules.

Example:

\(V_N = \{<\text{Sentence}>\}, <\text{noun}> <\text{verb}> <\text{adverb}>\}

\(\Sigma = \{\text{Ram, Sam, Ade, Ran}\}\)

\(S = <\text{Sentence}>\)

\(P = P_1\)
Example: For the grammar $G$ defined by the productions $1000$ (iii)

$$S \rightarrow AB \mid A1B \mid BA1$$
$$A \rightarrow 0A1S$$
$$B \rightarrow 0B \mid 1B1$$

Find the parse tree for the yields:

1. $1001$
2. $00101$
3. $00011$

Solution:

1) $1001$

```
S → AB
  ↓
  B
  ↓
  1B → 1B
  ↓
  101B → 1B
  ↓
  1001B → B
  ↓
  S
```

2) $00101$

```
S → AB
  ↓
  A
  ↓
  0A1B
  ↓
  A1B
  ↓
  1B
  ↓
  B
  ↓
  S
```
Properties:

1) The root node is always a node indicating start symbol.
2) The derivation is read from left to right.
3) The leaf nodes are always terminal nodes.
4) The interior nodes are always the non-terminal nodes.

Example: Draw a derivation tree for the string aabaaba for the CFG given by G, where

\[ P = \{ S \rightarrow aSa, \]
\[ S \rightarrow bSb, \]
\[ S \rightarrow aabaaba \]
Example:

The grammar \( G = (\{S, \{a, b\}, S, P\}) \) with productions

\[
S \to aSa, \quad S \to bSb, \quad S \to \lambda \text{ is context free.}
\]

\[
S \to aSa \\
\downarrow \\
aasaa \\
\downarrow \\
aab Sbaa \\
\downarrow \\
\lambda
\]

\[
\Rightarrow aabbaa
\]

\[
L(a) = \{ww^R \mid w \in \{a, b\}^+\}.
\]

This language is context free.

Example: Construct a grammar generating \( L = \omega c \omega^T \) where \( \omega \in \{a, b\}^* \).

Answer: The string which can be generated for given

\[
L \equiv \{aaca, aaeaa, bcb, baab, ababa, \ldots\}
\]

The grammar could be

\[
S \to aSa \\
S \to bSb \\
S \to c.
\]

Thus any kind of string could be derived from the given.
Context free grammar:

CFG is a way of describing languages using recursive rules or substitution rules called productions.

It consists of 4 tuples \((V, T, P, S)\):

- **V** - Set of variables
- **T** - Set of terminals
- **P** - Set of productions
- **S** - Start symbol.

**Example:**

Given a grammar \(G = (V, T, P, S)\),

The set of rules \(P\) is:

- \(S \to aSb\)
- \(S \to SS\)
- \(S \to e\)

The grammar generates strings such as \(aabb\).
Derivations and Languages:

The derivation of a CFG (from the productions to derive a string) can be represented using trees known as derivation trees or S-tree. Thus, a tree is a synonym for “derivation tree” if S is the start symbol. The derivation trees are used in the compilation process of programming languages.

A grammar is used to describe a language by generating each string of that language in the following manner:

1. Write down the start variable. It is the variable on the left-hand side of the top rule unless specified otherwise.
2. Find a variable that is written down and a rule that starts with that variable. Replace the written down variable.
Rightmost Derivation:

The rightmost derivation is a derivation in which the rightmost non-terminal is replaced first from the sentential form.

Example: \( S \rightarrow xyx \)
\( x \rightarrow 0 \)
\( y \rightarrow 1 \)

Answer:
\( S \rightarrow xyx \)
\( xy \rightarrow o (x \rightarrow o) \)
\( x \rightarrow 0 (y \rightarrow 1) \)
\( S \rightarrow 010 (x \rightarrow 0) \).

The derivation is starting from the right side.

Example: Let \( G \) be the Grammar \( S \rightarrow 0B | 1A, A \rightarrow 010B | 1AA, B \rightarrow 1 | 1B | 1BB \). For the string \( 001010 \), find in kappa notation derivation and derivation trees.

The given string:
\( S \rightarrow 0B | 1A \)
\( A \rightarrow 010B | 1AA \)
\( B \rightarrow 1 | 1B | 1BB \)

The string \( 001010 \).
A also represent as

\[ S \rightarrow T_1 T_2 T_3 T_n \]

This proves that \( S \Rightarrow S_1, S_2, \ldots, S_n \Rightarrow \alpha \) can be obtained.

Left-most and Right-most Derivation:

Derivation means replacement of non-terminal into terminal.

There are two types of derivation:

Left-most Derivation

Right-most Derivation

Left-most Derivation is a derivation in which the leftmost non-terminals is replaced first from the sentential form.

Example: The given string is

\[ S \rightarrow xyx \]
\[ x \rightarrow a \]
\[ y \rightarrow b \]

Solution:

\[ S \rightarrow xyx \]
\[ \Rightarrow ayy \ (X \rightarrow a) \]
\[ \downarrow \]
\[ abx \ (S \rightarrow S_0) \]
Relationship between Derivation and Derivation Tree:

**Theorem 1:** Let $G = (V, T, P, S)$ be a CFG. Then prove that $S \Rightarrow x \ y$ and only if there is a derivation tree in grammar $G$ with field $x$.

**Proof:** For a non-terminal and there exists $S \Rightarrow x \ y$ and only if there is a derivation tree starting from non-$S$ and yielding $x$.

To prove this we will use method of induction.

**Basis of induction:** Assume that there is only one interior node $S$.

The derivation tree yielding $S_1, S_2 \ldots S_n$.

From $S$ that means $S \Rightarrow S_1, S_2 \ldots S_n \Rightarrow x$ is input string.

![Derivation Tree Diagram]

**Induction Hypothesis:** we assume that for $k-1$ nodes the derivation tree can be drawn. We then can prove that for $k$ vertices also we can have a derivation tree that means the input string $x$ can be derived.

$S \Rightarrow S_1 S_2 \ldots S_k$

There are two cases: either $S_i$ may be a leaf variable $(\alpha_i)$ $S_i$ may be an interior yielding $x$. The $S$ denotes $x$ by fewer number of $k$ steps than $\alpha \in \mathcal{L}(S_1 S_2 S_3 \ldots S_k)$.

$\text{If } \alpha_i = S_i \text{ then } S_i \text{ is leaf node (terminal) and if } S_i \Rightarrow \alpha_i \text{ then } S_i$
Right most derivation:

$S \rightarrow T \rightarrow O \rightarrow T \rightarrow O \rightarrow T$ (1) $S \rightarrow T \rightarrow O \rightarrow T$

$T \rightarrow O \rightarrow T$ (2) $T \rightarrow O \rightarrow T$

$T \rightarrow O \rightarrow T$ (3) $T \rightarrow O \rightarrow T$

$T \rightarrow O \rightarrow T$

Right most derivation derived.

Ambiguity:

Ambiguity is defined as if there exists more than one parse trees for a given grammar, that means there could be more than one leftmost or rightmost derivation possible and then that grammar is said to be ambiguous grammar.

Example: Consider grammar by $G(V,T,P)$

$V = \{E, B\}$, $T = \{id\}$, $P = \{E \rightarrow E + E, E \rightarrow E * E, E \rightarrow id\}$

Selection:

LMD:

i) $E \rightarrow E + E$

$E \rightarrow id + E \rightarrow E + E$ (E $\rightarrow$ id),

$E \rightarrow id + E \rightarrow E + E$ (E $\rightarrow$ E$+$E)

$E \rightarrow id + id \rightarrow E + E$ (E $\rightarrow$ id),

$E \rightarrow id + id \rightarrow E + E$ (E $\rightarrow$ id$+$id)

STUDENTSFOCUS.COM
5. Rightmost derivation:

\[ S \rightarrow aB \]  
\[ \downarrow \]  
\[ aBB \]  
\[ \downarrow \]  
\[ aBB \]  
\[ \downarrow \]  
\[ aBbba \]  
\[ \downarrow \]  
\[ aBbba \]  
\[ \downarrow \]  
\[ aBBbba \]  
\[ \downarrow \]  
\[ aBbba \]  
\[ \downarrow \]  
\[ aBbba \]

Rightmost derivation is derived.

Example: Derive the string 1000 for leftmost and rightmost derivation using CFG 61: \[ \{V, T, P, S\} \] where

\[ V = \{S, T\} \]
\[ T = \{0, 1\} \]
\[ P = \{S \rightarrow T00T, T \rightarrow 01/1T1S\} \]

Answer: Leftmost derivation

\[ S \rightarrow T00T \]  
\[ (S \rightarrow T00T) \]
\[ \downarrow \]  
\[ T00T \]  
\[ (T \rightarrow 01) \]
\[ \downarrow \]  
\[ 01 \]
\[ \downarrow \]  
\[ 001 \]
\[ \downarrow \]  
\[ 001 \]
\[ \downarrow \]  
\[ 1000 \]
Example: "Derive the string "aabbb" for both leftmost and rightmost derivations using the above CFS given by

\[ S \rightarrow aB \mid bA \]
\[ A \rightarrow aA \mid bAA \]
\[ B \rightarrow b \mid bS \mid aBB \]

Answer:
1. If we want to derive leftmost derivation means start with leftmost productions.
2. If we want to derive rightmost derivation means start with rightmost productions.
3. Leftmost derivation:

\[ S \rightarrow aB \rightarrow aAB \rightarrow aabbB \rightarrow aabbBbS \rightarrow aabbBbSbA \rightarrow aabbBbSbAaA \]

\[ S \rightarrow bA \rightarrow bAA \rightarrow aaBB \rightarrow aaBBbS \rightarrow aaBBbSbA \rightarrow aaBBbSbAaA \]

STUDENTSFOCUS.COM
Left most Derivation:

\[ S \rightarrow 0B \]
\[ \downarrow \]
\[ 00BB \quad (B \rightarrow 0BB) \]
\[ \downarrow \]
\[ 00IB \quad (B \rightarrow 1) \]
\[ \downarrow \]
\[ 0011B \quad (8 \rightarrow 18) \]
\[ \downarrow \]
\[ 00110B \quad (8 \rightarrow 0B) \]
\[ \downarrow \]
\[ 0011018 \quad (B \rightarrow 18) \]
\[ \downarrow \]
\[ 00110108 \quad (B \rightarrow 0B) \]
\[ \downarrow \]
\[ 00110101 \quad (B \rightarrow 1). \]

From the given string we derived for leftmost derivation and derivation tree.
Properties of Context-free Languages

Three ways to simplify/clean a CFG
1. Eliminate *useless symbols (simplify)*
2. Eliminate $\epsilon$-productions
3. Eliminate unit productions

Eliminating useless symbols
A symbol $X$ is *reachable* if there exists:

$$S \Rightarrow^* \alpha X \beta$$

A symbol $X$ is *generating* if there exists:

$$X \Rightarrow^* w,$$

for some $w \in T^*$

For a symbol $X$ to be “useful”, it has to be both reachable and generating

$$S \Rightarrow^* \alpha X \beta \Rightarrow^* w', \text{ for some } w' \in T^*$$

Algorithm to detect useless symbols
1. First, eliminate all symbols that are not generating
   Next, eliminate all symbols that are not reachable

Example: Useless symbols

$$S \Rightarrow AB | a$$

$$A \Rightarrow b$$

1. $A$, $S$ are generating
2. $B$ is not generating (and therefore $B$ is useless)
3. ==> Eliminating $B$… (i.e., remove all productions that involve $B$)
   1. $S \Rightarrow a$
   2. $A \Rightarrow b$
4. Now, $A$ is not reachable and therefore is useless
5. Simplified G:
   1. $S \Rightarrow a$

Eliminating $\epsilon$-productions

Caveat: It is *not* possible to eliminate $\epsilon$-productions for languages which include $\epsilon$ in their word set

Theorem: If $G=(V,T,P,S)$ is a CFG for a language $L$, then $L\setminus \{\epsilon\}$ has a CFG without $\epsilon$-productions

*Definition:* $A$ is “nullable” if $A \Rightarrow^* \epsilon$

If $A$ is nullable, then any production of the form “$B \Rightarrow CAD$” can be simulated by:

$$B \Rightarrow CD | CAD$$

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

Algorithm:
1. Detect all nullable variables in $G$
2. Then construct $G_1=(V,T,P_1,S)$ as follows:
   i. For each production of the form: $A \Rightarrow X_1X_2…X_k$, where $k \geq 1$, suppose $m$ out of the $k$ $X_i$’s are nullable symbols
   ii. Then $G_1$ will have $2^m$ versions for this production
i. i.e, all combinations where each X_i is either present or absent

iii. Alternatively, if a production is of the form: A \rightarrow \epsilon, then remove it

Eliminating unit productions
Unit production is one which is of the form A \rightarrow B, where both A & B are variables

E.g.,

iv. \ E \rightarrow \ T \mid \ E+T

v. \ T \rightarrow \ F \mid \ T*F

vi. \ F \rightarrow \ I \mid \ (E)

vii. \ I \rightarrow \ a \mid \ b \mid Ia \mid Ib \mid I0 \mid I1

How to eliminate unit productions?

viii. Replace E \rightarrow T with E \rightarrow F \mid T*F

ix. Then, upon recursive application wherever there is a unit production:

i. \ E \rightarrow F \mid T*F \mid E+T \quad \text{(substituting for T)}

ii. \ E \rightarrow I \mid (E) \mid T*F \mid E+T \quad \text{(substituting for F)}

iii. \ E \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \mid (E) \mid T*F \mid E+T \quad \text{(substituting for I)}

iv. Now, E has no unit productions

Similarly, eliminate for the remainder of the unit productions

Chomsky Normal Form

A context-free grammar G = (V, Σ, R, S) is said to be in Chomsky Normal Form (CNF), if and only if every rule in R is of one of the following forms:

1. A \rightarrow a, for some A \in V and some a \in \Sigma

2. A \rightarrow BC, for some A \in V and B, C \in V \{S\}

3. S \rightarrow \epsilon

In other words: Every rule either replaces a variable by a single character or by a pair of variables except the start symbol, and the only rule that can have the empty word as its right-hand side must have the start symbol as its left-hand side. From the above definition it follows, that every parse tree for a grammar in CNF must be a binary tree, and the parse tree for any non-empty word cannot have any leaves labeled with \epsilon in it. The use for the Chomsky Normal Form is to make many of the proofs about context-free languages we will encounter later much easier by allowing us to assume that every context-free grammar we want to reason about is in Chomsky Normal Form. We will first see the usefulness of CNF in the proof for the Context-Free Pumping Lemma.
UNIT-II CONTEXT FREE GRAMMARS AND LANGUAGES

PART-A

1. Define CFG. Find \( L(G) \) where \( G = (\{ S \}, \{ 0, 1 \}, \{ S -> 0S1, S -> \epsilon \}, S) \).
2. Define derivation tree for a CFG (or) Define parse tree.
3. Construct the CFG for generating the language \( L = \{ a^n b^n | n \geq 1 \} \).
4. Let \( G \) be the grammar \( S -> aB/bA, A -> a/aS/bA, B -> b/bS/aB \). Find the derivation tree for the string \( aababbba \).
5. Let \( G \) be the grammar \( S -> aB/bA, A -> a/aS/bA, B -> b/bS/aB \). Obtain parse tree for the string \( aababbba \).
6. For the grammar \( S -> aCa, C -> aCa/b \), find \( L(G) \).
7. Show that \( id + id^* id \) can be generated by two distinct leftmost derivations in the grammar \( E -> E + E | E^* E | (E) | id \).
8. For the grammar \( S -> A1B, A -> 0A/\epsilon, B -> 0B/1B/\epsilon \), give leftmost and rightmost derivations for the string \( 00101 \).
9. Find the language generated by the CFG \( G = (\{ S \}, \{ 0, 1 \}, \{ S -> 0S0/1S1, S -> \epsilon \}) \).
10. Obtain the derivation tree for the grammar \( G = (\{ S, A \}, \{ a, b \}, P, S) \) where \( P \) consist of \( S -> aAS/a, A -> SBa/SS/ba \).
11. Consider the alphabet \( \Sigma = \{ a, b, (,), +, *, \epsilon \} \). Construct the context free grammar that generates all strings in \( \Sigma^* \) that are regular expression over the alphabet \( \{ a, b \} \).
12. Write the CFG to generate the set \( \{ a^m b^n c^p | n = p \geq 1 \} \).
13. Construct a derivation tree for the string \( 0011000 \) using the grammar \( S -> A0S | 0S | SS, A -> S1A | 10 \).
14. Give an example for a context free grammar.
15. Let the production of the grammar be \( S -> 0B/1A, A -> 0S/1A, B -> 1/1S/0B \). For the string \( 0110 \) find the rightmost derivation.
16. What is the disadvantage of unambiguous parse tree? Give an example.

PART-B

1. a) Let \( G \) be a CFG and let \( a \rightarrow w \) in \( G \). Then show that there is a leftmost derivation of \( w \).

b) Let \( G = (V, T, P, S) \) be a Context free Grammar then prove that if \( S \rightarrow \alpha \) then there is a derivation tree in \( G \) with yield \( \alpha \).

2. Let \( G \) be a grammar \( s \rightarrow OB/1A, A \rightarrow O/OS/1AA, B \rightarrow 1/1S/0B \). For the string \( 00110101 \) find its leftmost derivation and derivation tree.

3. a) If \( G \) is the grammar \( S \rightarrow Sbs/a \), Show that \( G \) is ambiguous.
b) Give a detailed description of ambiguity in Context free grammar

4. a) Show that \( E \rightarrow E + E / E ^{*} / E / id \) is ambiguous. (6) b) Construct a Context free grammar \( G \) which accepts \( N(M) \), where \( M = \{ q_0, q_1, \{ a, b \}, \{ z_0, z \}, \delta, q_0, z_0, \emptyset \} \) and where \( \delta \) is given by

\[
\delta(q_0, b, z_0) = \{(q_0, zz_0)\}
\]

\[
\delta(q_0, \varepsilon, z_0) = \{(q_0, \varepsilon)\}
\]

\[
\delta(q_0, b, z) = \{(q_0, zz)\}
\]

\[
\delta(q_0, a, z) = \{(q_1, z)\}
\]

\[
\delta(q_1, b, z) = \{(q_1, \varepsilon)\}
\]

\[
\delta(q_1, a, z_0) = \{(q_0, z_0)\}
\]

5. a) If \( L \) is Context free language then prove that there exists PDA \( M \) such that \( L = N(M) \).

b) Explain different types of acceptance of a PDA. Are they equivalent in sense of language acceptance? Justify your answer.

6. Construct a PDA accepting \( \{ a^n b^m a^n \mid m,n \geq 1 \} \) by empty stack. Also construct the corresponding context-free grammar accepting the same set.

7. a) Prove that \( L \) is \( L(M_2) \) for some PDA \( M_2 \) if and only if \( L \) is \( N(M_1) \) for some PDA \( M_1 \).

b) Define deterministic Push Down Automata DPDA. Is it true that DPDA and PDA are equivalent in the sense of language acceptance is concern? Justify your answer.

8. a) Construct a equivalent grammar \( G \) in CNF for the grammar \( G_1 \) where \( G_1 = \{ S, A, B \}, \{ a, b \}, \{ S \rightarrow bA/aB, A \rightarrow bAA/aS/a, B \rightarrow aBB/bS/b \}, S \} \)
b) Find the left most and right most derivation corresponding to the tree.

9. a) Find the language generated by a grammar

\[ G = (\{S\}, \{a,b\}, \{S \rightarrow aSb, S \rightarrow ab\}, S) \] (4)

b) Given \( G = (\{S,A\}, \{a,b\}, P, S) \) where \( P = \{ S \rightarrow AaS | SS, A \rightarrow SbA | ba \} \)

\( S \)-Start symbol. Find the left most and right most derivation of the string \( w = aabbaaa \). Also construct the derivation tree for the string \( w \).

c) Define a PDA. Give an Example for a language accepted by PDA by empty stack.

10. \( G \) denotes the context-free grammar defined by the following rules. \( S \rightarrow ASB | ab | SS \) \( A \rightarrow aA | A \), \( B \rightarrow bB | A \)

(i) Give a left most derivation of \( aaabb \) in \( G \). Draw the associated parse tree.

(ii) Give a right most derivation of \( aaabb \) in \( G \). Draw the associated parse tree.

(iii) Show that \( G \) is ambiguous. Explain with steps.

(iv) Construct an unambiguous grammar equivalent to \( G \). Explain.