5. CONTEXT - FREE LANGUAGES (CFL)

CHAPTER - 5
CONTEXT - FREE LANGUAGES (CFL)

In this chapter we study the concept of context free grammars and languages. We further define derivation trees, ambiguity, relationship between derivation and derivation trees with examples.

5.1 CONTEXT-FREE GRAMMAR (CFG)

A CFG is a way of describing languages by *recursive rules* (or) *substitution rules* called *productions*. A CFG consists of a set of *variables*, a set of *terminal symbols*, and a *start variable* as well as the *productions*. Each production consists of a head variable and a body consisting of a string of zero or more variables and / or terminals.

Note :

Variable symbol	-	represented by capital letters
Terminal symbol	-	represented by lower case letters, numbers or special symbols.
Start variable	-	occurs on the left-hand side of the topmost rule.

Example 5.1

Grammar G1

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

No. of production rules	: 3
Variables	: A, B
Start symbol	: A
Terminals	: 0,1,#

5.1.1 Definition of Context-Free Grammar (CFG)

A *Context-Free Grammar* (CFG) is denoted by G=(V, T, P, S), where V and T are *variables* and *terminals* respectively. We assume that V and T are disjoint. P is a finite set of *productions*; each production is of the form $A \rightarrow \alpha$, where A is a variable and α is a string of symbols from $(V \cup T)^*$. S is a special variable called the *start symbol*.

Example 5.2

G = ({E}, {+, *, (,), *id*}, P, E), where P consists of :

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow (E)$

$E \rightarrow id$

The above productions of the the form $A \rightarrow \alpha$ can be rewritten as list of productions of grammar G

(i.e.) $A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots\ldots \mid \alpha_k$

$E \rightarrow E+E \mid E \times E \mid (E) \mid id$, where vertical line denotes (OR)

Note :

V_N (or) V	-	Nonterminals or variables
Σ (or) T	-	Terminals
λ (or) ε	-	Empty string
$\underset{G}{\Rightarrow}$	-	Production rule is applied only once to derive a terminal string from the start symbol.
$\overset{*}{\underset{G}{\Rightarrow}}$	-	Production rules are applied more than once to derive a terminal string from the start symbol (i.e.) reflexive and transitive closure of G.
$S \overset{*}{\Rightarrow} \alpha$	-	Sentential form ($\alpha \rightarrow (V \cup T)^*$)

5.1.2 Definition of Context Free Language (CFL)

The language generated by CFG G is defined as :

$L(G) = \{w \mid w$ is in T^+ and $S \overset{*}{\underset{G}{\Rightarrow}} w\}$. That is a string is in L(G) if: :

(i) The string consists of terminals only

(ii) The string has to be derived from S only

L is a Context Free Language (CFL), if it is L(G) for some CFG G.

Example 5.3

Let CFL be the set of all palindromes over $\{a, b\}$. Construct CFG generating CFL.

Solution :

For constructing a grammar (CFG) generating a set of all palindromes, we use the recursive definition:

(a) ε, a and b are palindromes.

(b) if w is palindrome awa, bwb are palindromes.

(c) Nothing else is a palindrome.

$\therefore$ the set P is defined as:

$$S \rightarrow \varepsilon \mid a \mid b$$

$$S \rightarrow aSa \mid bSb$$

(i.e.) Let G = ({S}, {a,b}, P, S). Then

$$S \Rightarrow \varepsilon,\ S \Rightarrow a,\ S \Rightarrow b,\ S \overset{*}{\Rightarrow} aSa,\ S \overset{*}{\Rightarrow} bSb$$

$\therefore$ ε, a, b, $w \in$ L(G)

(i.e.) L = L(G)

5.1.3 Applications of CFG

(i) ***Specification and compilation of programming languages:*** A grammar for a programming language often appears as a reference for people trying to learn the language syntax. Designers of compilers, interpreters for programming languages often start by obtaining a grammar for the language to design a parser. Ex : YACC

(ii) ***Document Type Definitions (DTD):*** The emerging XML standard for sharing information through web documents has a notation, called the DTD, for describing the structure of such documents, through the nesting of semantic tags within the document. The DTD is in a context-free grammar whose language is a class of related documents.

5.2 DERIVATIONS AND LANGUAGES

The derivation of a CFG (from the productions to derive a strings) can be represented using trees known as *derivation trees* or *parse trees* or *s-trees*. Thus s-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:

(i) Write down the start variable. It is the variable on the left-hand side of the top rule, unless specified otherwise.

(ii) Find a variable that is written down and a rule that starts with that variable. Replace the written down variable with the right hand side of that rule.

(iii) Repeat step (ii) until no variables remain.

The sequence of substitutions to obtain a string is called a *derivation.*

5.2.1 Definition of derivation tree

Let G=(V, T, P, S) be a CFG. A tree is a *derivation* (or) *parse tree* for G if :

(i) Every vertex has a label which is a variable (or) terminal (or) λ, (i.e.) $V \cup T \cup (\lambda)$.

(ii) The label of the root is S(Start symbol)

(iii) The internal vertices must be in V (variable) labeled as A.

(iv) If n has label A and vertices $n_1, n_2, \ldots\ldots\ldots n_k$ are the sons of vertex n, in order from the left, with labels $x_1, x_2 \ldots\ldots x_k$ respectively, then $A \rightarrow x_1 x_2 \ldots\ldots\ldots x_k$ must be a production in P.

(v) If vertex n has label λ, then n is a leaf and is the only son of its father.

Example 5.4

Consider the grammar G = ({S, A}, {*a*,*b*}, P,S), where P consists of

$S \rightarrow aAS \mid b$

$A \rightarrow SbA \mid ba$

Draw its equivalent derivation tree for $w = abbbab$

(i) The vertices are numbered for reference (i.e.) 1, 2....11

(ii) The label of the vertices are variables (or) terminals.

(iii) The label of the root vertex is S start symbol

(iv) The interior vertices are 1, 3, 4, 5, 7 which are variables.

(v) Vertex 1 has label S, and its sons from the left, have labels *a*, A and S therefore $S \rightarrow aAs$ is a production similarly for vertex 3 : $A \rightarrow SbA$.

Vertices 4 and 5 : S $\rightarrow a$

Vertex 7 : A $\rightarrow ba$ are the productions.

(vi) Thus the conditions (i)–(v) satisfies the constraints of a derivation tree for the given G.

(vii) Thus the left-to-right ordering of derivation is : $S \stackrel{*}{\Rightarrow} aAS \stackrel{*}{\Rightarrow} aSbAb \stackrel{*}{\Rightarrow} abbbab$

Example 5.5

Derivation of the string w=000#111 from grammar G1

A $\Rightarrow$ 0A1

$\Rightarrow$ 00A11 (A$\rightarrow$0A1)

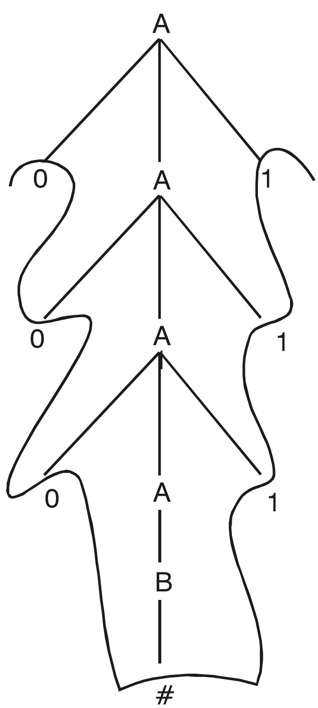
$\Rightarrow$ 000A111 (A$\rightarrow$0A1)

$\Rightarrow$ 000B111 (A$\rightarrow$B)

$\Rightarrow$ 000#111 (B$\rightarrow$#)

Parse tree for 000#111 from grammar G1

All strings generated in this way contribute the language of the grammar (i.e.) Context Free Language (CFL).



Example 5.6

Grammar G2

Fragment of the English language

<SENTENCE> → <NOUN-PHRASE><VERB PHRASE>

<NOUN-PHRASE> → <CMPLX-NOUN>|<PREP-PHRASE>

<VERB-PHRASE> → <CMPLX-VERB>|<PREP-PHRASE>

<PREP-PHRASE> → <PREP> <CMPLX-NOUN>

<CMPLX-NOUN> → <ARTICLE><NOUN>

<CMPLX-VERB> → <VERB>|<VERB><NOUN-PHRASE>

<ARTICLE> → a | the | an

<NOUN> → boy | girl | flower

<VERB> → touches | likes | sees

<PREP> → with

Strings that can be derived from grammar G2 are:

the boy sees a flower

a boy sees

a girl with a flower

Derivation of the first string from the above list :

<SENTENCE> ⇒ <NOUN-PHRASE> <VERB-PHRASE>

⇒ <CMPLX-NOUN> <VERB-PHRASE>

⇒ <ARTICLE> <NOUN> <VERB-PHRASE>

⇒ the boy < VERB-PHRASE>

⇒ the boy <CMPLX - VERB>

⇒ the boy <VERB> <NOUN-PHRASE>

⇒ the boy sees <CMPLX-NOUN>

⇒ the boy sees <ARTICLES><NOUN>

⇒ the boy sees a flower

Derivation of the second string from the above list

<SENTENCE> ⇒ <NOUN PHRASE> <VERB PHRASE>

⇒ <CMPLX-NOUN> <VERB-PHRASE>

⇒ <ARTICLE> <NOUN><VERB-PHRASE>

⇒ a <NOUN> <VERB-PHRASE>

⇒ a boy <VERB-PHRASE>

⇒ a boy <CMPLX-VERB>

⇒ a boy <VERB>

⇒ a boy sees.

Example 5.7

Grammar G3

Context-free grammar with Backus-Naur Form (BNF) representation

(i) <expression> → <expression> + <expression>

(ii) <expression> → <expression>   <expression>

(iii) <expression> → (<expression>)

(iv) <expression> → id

variable – <expression>

terminals – +,   , (,), id.

Derivation of string (id+id)   id from the above production

<expression> ⇒ <expression>   <expression>

⇒ (<expression>)   <expression>

⇒ (<expression>)   id

⇒ (<expression> + <expression>)   id

⇒ (<expression> + id)   id

⇒ (id+id)   id

5.2.2 Subtree of a derivation tree

A *subtree* of a derivation tree T is a tree satisfying the following constraints :

(a) Whose root is some vertex *v* of V.

(b) Whose vertices are the descendants of *v* together with their labels.

(c) Whose edges are those connecting the descendants of *v*.

Example 5.8

A substree (A $\stackrel{*}{\Rightarrow}$ *bbba*) (A ⇒ *ba*) which is derived from the above discussed derivation tree (S $\stackrel{*}{\Rightarrow}$ *abbbab*) - Example 5.4.

A subtree looks like a derivation tree except that the label of the root may not be S. It is called an A-*tree*, if the label of the root is A.

5.2.3 Leftmost and Rightmost derivation

Leftmost derivation

A derivation $A \overset{*}{\Rightarrow} w$ is called a *leftmost derivation* if we apply a production only to the *leftmost variable* at every step.

Rightmost derivation

A derivation $A \overset{*}{\Rightarrow} w$ is called a *rightmost derivation* if we apply a production only to the *rightmost variable* at every step.

Example 5.9

Consider G whose productions are $S \rightarrow aAS \mid a$, $A \rightarrow SbA \mid SS \mid ba$. For the string $w = aabbaa$ find :

(a) Leftmost derivation

(b) Rightmost derivation

(c) Derivation tree

Solution :

(a) Leftmost derivation

$$\begin{aligned} S &\Rightarrow aAS \\ &\Rightarrow aSbAS \quad (A \rightarrow SbA) \\ &\Rightarrow aabAS \quad (S \rightarrow a) \\ &\Rightarrow aabbaS \quad (A \rightarrow ba) \\ &\Rightarrow aabbaa \quad (S \rightarrow a) \end{aligned}$$

(i.e.) $S \overset{*}{\Rightarrow} a^2b^2a^2$ - at each step the production rule is applied to the leftmost variable.

(b) Rightmost derivation

S $\Rightarrow aAS$

$\Rightarrow aAa(S\rightarrow a)$

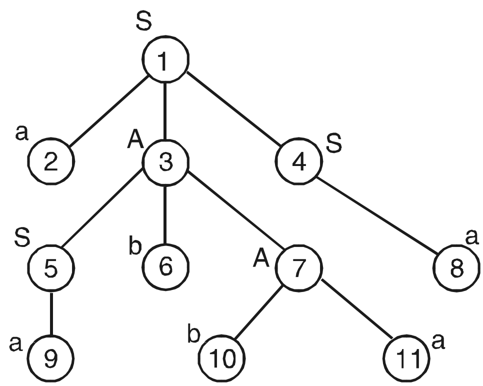
$\Rightarrow aSbAa(A\rightarrow SbA)$

$\Rightarrow aSbbaa(A\rightarrow ba)$

$\Rightarrow aabbaa(S\rightarrow a)$

(i.e.) $S\overset{*}{\Rightarrow}a^2b^2c^2$ - at each step the production rule is applied to the rightmost variable.

(c) Derivation tree



Theorem : If $A\overset{*}{\Rightarrow}w$ in G, then there is a leftmost derivation of w.

Proof

Basis

A⇒w is a leftmost derivation as L.H.S. as only one variable.

Induction

$A\overset{*}{\Rightarrow}w$ can be derived in atmost k step (i.e.) $A\Rightarrow x_1x_2......x_m\overset{k}{\Rightarrow}w$.

The string w can be split as $w_1w_2.......w_m$ such that $x_i=w_i$. As $x_i\overset{*}{\Rightarrow}w_i$ involves atmost k steps by induction hypothesis, the leftmost derivation of w is:

$A\Rightarrow x_1\ x_2........\ x_m \overset{*}{\Rightarrow} w_1x_2...........\ x_m \overset{*}{\Rightarrow} w_1\ w_2\ x_3.......x_m$

$\overset{*}{\Rightarrow} w_1w_2......w_m$

Hence by induction the result is true for all derivations $A\overset{*}{\Rightarrow}w$.

Corollary : Every derivation tree of w induces a leftmost derivation of w.

5.3 THE RELATIONSHIP BETWEEN DERIVATION TREES AND DERIVATIONS

Theorem

Let G = (V_N, Σ, P, S) be a context free grammar (CFG). Then $S \overset{*}{\Rightarrow} \alpha$ if and only if there is a derivation tree for G which yield α.

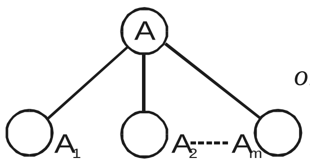
Proof

Step 1:

We prove that $A \overset{*}{\Rightarrow} \alpha$ if and only if there is an A-tree which derives α. Once this is proved, the theorem follows by assuming that A=S.

Let α be the yield of an A-tree T. We prove that $A \overset{*}{\Rightarrow} \alpha$ by induction on the number of internal vertices in T.

When the tree has only one internal vertex, the remaining vertices are leaves and are the sons of the root.



A tree with only one internal vertex

By the definition of derivation tree (iv) $A \rightarrow A_1\ A_2 .... A_m = \alpha$ is a production in G (i.e.) $A \Rightarrow \alpha$. This is a basis step for induction (k=1). Now assume the result is true for k–1 internal vertices (k>1).

Let T be an A-tree with k internal vertices ($k \geq 2$). Let $v_1, v_2, ... v_m$ be the sons of the root in the left-to-right ordering. Let their labels be $x_1, x_2 .... x_m$. By the definition of derivation tree (iv) $A \rightarrow x_1\ x_2 ..... x_m$ is one of the production P. Therefore:

$A \Rightarrow x_1, x_2 .... x_m$

As $k \geq 2$, at least one of the sons is an internal vertex. By the left-to-right ordering of leaves, α can be written as $\alpha_1, \alpha_2, ... \alpha_m$, where α_i is obtained by :

(a) The concatenation of labels of the leaves which are descendents of vertex v_i. v_i is an internal vertex of the subtree $x_i \overset{*}{\Rightarrow} \alpha_i$

(b) If v_i is not an internal vertex (i.e.) a leaf, then $x_i = \alpha_i$

$\therefore A \overset{*}{\Rightarrow} x_1\ x_2 ..... x_m \overset{*}{\Rightarrow} \alpha_1\ x_2 .... x_m.$

$\overset{*}{\Rightarrow} \alpha_1\ \alpha_2 .... \alpha_m = \alpha.$

(i.e.) $A \overset{*}{\Rightarrow} \alpha$ (by Induction Principle)

Step 2 :

To prove the "only if" part, let us assume that $A \overset{*}{\Rightarrow} \alpha$.

When $A \Rightarrow \alpha$, $A \rightarrow \alpha$ is a production in P. If $\alpha = x_1 x_2 .... x_m$, the A-tree with yield α is basis for induction. That is :

Assume the result for derivations in atmost k steps. Let $A \overset{k}{\Rightarrow} \alpha$, split this as :

$A \Rightarrow x_1\ x_2 ..... x_m \overset{k-1}{\Rightarrow} \alpha$. $A \Rightarrow x_1 ...... x_m$ implies

$A \Rightarrow x_1 x_2 .... x_m$ is a production in P.

In the derivation $x_1\ x_2 .... x_m \overset{k-1}{\Rightarrow} \alpha$, either

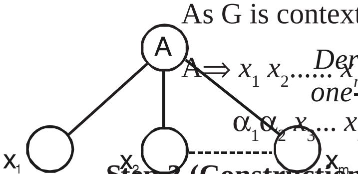
(a) x_i is not changed throughout the derivation (i.e.) $x_i = \alpha_i$

(b) x_i is changed in some subsequent step. (i.e.) $x_i \overset{*}{\Rightarrow} \alpha_i$

As G is context free, replace a single variable α by a string $\alpha_1 \alpha_2 ..... \alpha_m$

$A \Rightarrow x_1\ x_2 ...... x_m \overset{*}{\Rightarrow} \alpha_1\ x_2 ...... x_m \overset{*}{\Rightarrow}$

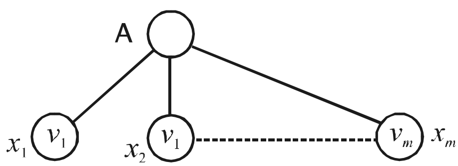
$\alpha_1 \alpha_2\ x_3 .... x_m \overset{*}{\Rightarrow} .... \overset{*}{\Rightarrow} \alpha_1 \alpha_2 ...... \alpha_m = \alpha$



Derivation tree for one-step derivation

Step 3 (Construction of derivation tree to derive α) :

As $A \rightarrow x_1\ x_2, ... x_m$ is in P, construct a tree with m leaves, shown below:



Derivation tree of $x_1 .... x_m$

(a) Vertex v_i is not changed (i.e.) $x_i = \alpha_i$, where x_i is terminal.

(b) $x_i \overset{*}{\Rightarrow} \alpha_i$ in less than k steps, if x_i is a variable.

If x_i is a variable, then the derivation of α_i from x_i must take fewer than k steps, since the entire derivation $A \overset{*}{\Rightarrow} \alpha$ takes k steps, and the first step $(x_i = \alpha_i)$ is surely not part of the derivation $x_i \overset{*}{\Rightarrow} \alpha_i$. Thus by inductive hyphothesis, for each x_i, that is a variable, there is an x_i tree with yield α_i. Let this tree be T_i.

In the above representation :

(a) Vertex labeled x_i is reptaced by T_i if it is not a terminal.

(b) If x_i is a terminal, no replacement is made.

(c) Therefore the yield of tree is α.

5.4 AMBIGUOUS GRAMMAR

A context free grammar G is said to be *ambiguous* if there exists some $w \in L(G)$ that has at least two distinct derivation trees. Alternatively, *ambiguity* implies the existence of two or more leftmost or rightmost derivations.

Example 5.10

Consider G = ({S}, {*a*,*b*,+,*}, P,S) where P consists of S→S+S | S * S | *a* | *b*. From the given G, P two leftmost derivations of *a*+*a*   *b* are induced. They are:

S ⇒ S+S

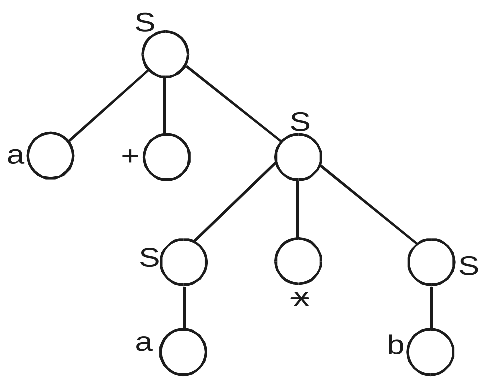
⇒ *a*+S (S→*a*)

⇒ *a*+S   S (S→S   S)

⇒ *a*+*a*   S (S→*a*)

⇒ *a*+*a*   *b* (S→*b*)

The corresponding derivation tree is :



S $\Rightarrow$ S ~~x~~ S

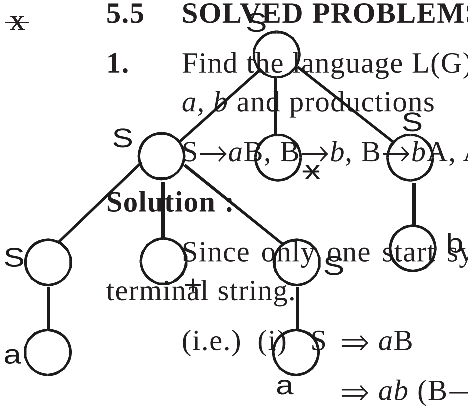
$\Rightarrow$ S + S S (S$\rightarrow$S+S)

$\Rightarrow$ a + S S (S$\rightarrow$$a$)

$\Rightarrow$ a + a S (S$\rightarrow$$a$)

$\Rightarrow$ a + a b (S$\rightarrow$$b$)

The corresponding derivation tree is :



Therefore a + a b is ambigious.

5.5 SOLVED PROBLEMS

1. Find the language L(G) generated by the grammar G with variables S, A, B terminals a, b and productions

S$\rightarrow$$aB, B\rightarrow$$b$, B$\rightarrow$$bA, A\rightarrow$$a$B

Solution :

Since only one start symbol is given, apply the productions to observe the form of terminal string.

(i.e.) (i) S $\Rightarrow$ aB

$\Rightarrow$ ab (B$\rightarrow$$b$)

(ii) S $\Rightarrow$ aB

$\Rightarrow$ abA (B$\rightarrow$$b$A)

$\Rightarrow$ abaB (A$\rightarrow$$a$B)

$\Rightarrow$ $abab$ (B$\rightarrow$$b$)

(iii) S $\Rightarrow$ aB

$\Rightarrow$ abA (B$\rightarrow$$b$A)

$\Rightarrow$ abaB (A$\rightarrow$$a$B)

$\Rightarrow$ $abab$A (B$\rightarrow$$b$A)

$\Rightarrow ababaB$ ($A \to aB$)

$\Rightarrow ababab$ ($B \to b$)

$L(G) = \{(ab)^n = abab..... ab : n \geq 1\}$

2. If G is a grammar $S \to sba|a$ prove that G is ambiguous.

Solution : **(Apr/May 2004)**

Let $w = abababa$

Leftmost derivations

(i) $S \Rightarrow SbS \Rightarrow abS \Rightarrow abSbS \Rightarrow ababS$

$\Rightarrow ababSbS \Rightarrow abababS \Rightarrow abababa$

(ii) $S \Rightarrow SbS \Rightarrow SbSbS \Rightarrow abSbS \Rightarrow ababS$

$\Rightarrow ababSbS \Rightarrow abababS \Rightarrow abababa$

Derivation trees for w=abababa

For the string w=abababa two leftmost derivations are exist. Therefore the G is ambiguous.

3. Let G be the grammar $S \rightarrow 0B|1A$, $A \rightarrow 0|0S|1AA$, $B \rightarrow 1|1S|0BB$. For the string 00110101 find

(a) Leftmost derivation

(b) Rightmost derivation

(c) Derivation tree

(d) For the string 0110 find a rightmost derivation. **(Apr/May 2004)**

(May/Jun 2007)

Solution :

(a) *Leftmost derivation*

$S \Rightarrow 0B$

$\Rightarrow 00BB$ $(B \rightarrow 0BB)$

$\Rightarrow 001B$ $(B \rightarrow 1)$

$\Rightarrow 0011S$ $(B \rightarrow 1S)$

$\Rightarrow 00110B$ $(S \rightarrow 0B)$

$\Rightarrow 001101S$ $(B \rightarrow 1S)$

$\Rightarrow 0011010B$ $(S \rightarrow 0B)$

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

(b) *Rightmost derivation*

$S \Rightarrow 0B$

$\Rightarrow 00BB$ $(B \rightarrow 0BB)$

$\Rightarrow 00B1S$ $(B \rightarrow 1S)$

$\Rightarrow 00B10B$ $(S \rightarrow 0B)$

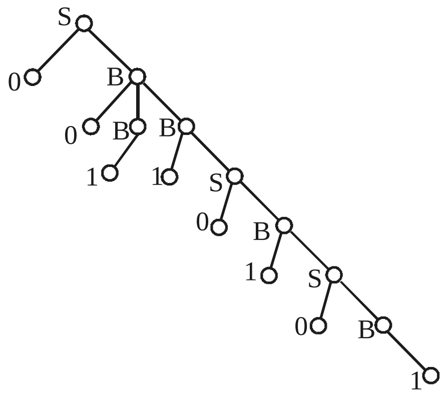
$\Rightarrow 00B101S$ $(B \rightarrow 1S)$

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

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

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

(c) *Derivation tree*



(d) $S \Rightarrow 0B$

$\Rightarrow 0llA \; (S \rightarrow 1A)$

$\Rightarrow 0ll0 \; (A \rightarrow 0)$

4. Consider the grammar $S \rightarrow aS|aSbS|\varepsilon$. This grammar is ambiguous. Show that the string *aab* has two

(a) Parse trees (b) Leftmost derivations (c) Rightmost derivations

Solution :

Parse trees :

Leftmost derivations

$S \Rightarrow aS \Rightarrow aaSbS \Rightarrow aabS \Rightarrow aab$ and

$S \Rightarrow aSbS \Rightarrow aaSbS \Rightarrow aabS \Rightarrow aab$

Rightmost derivations

$S \Rightarrow aS \Rightarrow aaSbS \Rightarrow aaSb \Rightarrow aab$ and

$S \Rightarrow aSbS \Rightarrow aSb \Rightarrow aaSb \Rightarrow aab$

5. Let G be the grammar **(Nov./Dec 2004), (May/Jun 2007), (Apr/May 2008)**

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

For the string *aaabbabbba* find a leftmost derivation.

Solution :

$S \Rightarrow aB$

$\Rightarrow aaBB\ (B \rightarrow aBB)$

$\Rightarrow aaaBBB\ (B \rightarrow aBB)$

$\Rightarrow aaabBB\ (B \rightarrow b)$

$\Rightarrow aaabbB\ (B \rightarrow b)$

$\Rightarrow aaabbaBB\ (B \rightarrow aBB)$

$\Rightarrow aaabbabB\ (B \rightarrow b)$

$\Rightarrow aaabbabbS\ (B \rightarrow bS)$

$\Rightarrow aaabbabbbA\ (S \rightarrow bA)$

$\Rightarrow aaabbabbba\ (A \rightarrow a)$

6. Let G be the grammar $S \rightarrow aS \mid aSbS \mid \varepsilon$. Prove that **(Nov./Dec 2004)**

$L(G)=\{x|\text{each prefix of } x \text{ has atleast as many } a\text{'s as } b\text{'s}\}.$

Solution :

$S \Rightarrow aS$

$\Rightarrow aaSbS \quad (S \rightarrow aSbS)$

$\Rightarrow aaaSbS \quad (S \rightarrow aS)$

$\Rightarrow aaabbb \quad (S \rightarrow b)$

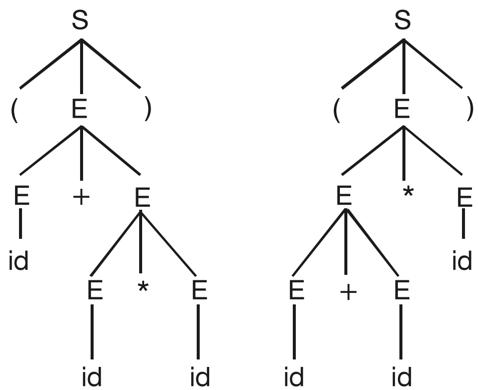
$\therefore$ x has atleast as many a's as b's.

7. Show that $E \rightarrow E + E \mid E * E \mid (E) \mid id$ is ambiguous. **(Apr/May 2005)** **(Nov./Dec 2005)**

Solution :

(i) $S \Rightarrow (E)$
$\Rightarrow (E+E)$
$\Rightarrow (id+E)$
$\Rightarrow (id+E*E)$
$\Rightarrow (id+id*E)$
$\Rightarrow (id+id*id)$

(ii) $S \Rightarrow (E)$
$\Rightarrow (E*E)$
$\Rightarrow (E+E*E)$
$\Rightarrow (id + E*E)$
$\Rightarrow (id+id*E)$
$\Rightarrow (id+id*id)$



From (i) & (ii) $(id+id*id)$ is ambiguous, because of two distinct trees.

8. For the grammar $S \rightarrow A1B, A \rightarrow 0A|\varepsilon, B \rightarrow 0B|1B|\varepsilon$, give left most and right most derivation of the following string 00101. **(May/Jun 2006)**

Solution :

(a) Leftmost derivation

$S \Rightarrow A1B$
$\Rightarrow 0A1B \ (A \rightarrow 0A)$
$\Rightarrow 00A1B \ (A \rightarrow 0A)$
$\Rightarrow 001B \ (A \rightarrow \varepsilon)$
$\Rightarrow 0010B \ (B \rightarrow 0B)$
$\Rightarrow 00101B \ (B \rightarrow 1B)$
$\Rightarrow 00101 \ (B \rightarrow \varepsilon)$

(b) Rightmost derivation

$S \Rightarrow A1B$

$\Rightarrow A10B\ (B \to 0B)$

$\Rightarrow A101B\ (B \to 1B)$

$\Rightarrow A101\ (B \to \varepsilon)$

$\Rightarrow 0A101\ (A \to 0A)$

$\Rightarrow 00A101\ (A \to 0A)$

$\Rightarrow 00101\ (A \to \varepsilon)$

9. Construct CFG to generate $\{a^n b^n \mid n \in Z^+\}$. **(May/Jun 2006)**

Solution :

$G = (\{S\}, \{a,b\}, p, S)$ where

$P = \{S \to aSb \mid ab\}$

$S \Rightarrow aSB$

$\Rightarrow aaSbb (S \to aSb)$

$\Rightarrow aaabbb (S \to ab)$

$R \Rightarrow RR$ $R \Rightarrow (R)$ $R \Rightarrow R^*$

$\Rightarrow a^n b^n$ for $n \geq 1$

10. Consider the alphabet $\Sigma = \{a, b, (\ ,\ ), +, *, ., \varepsilon\}$. Construct a context free grammar that generates all strings in Σ^* that are regular expressions over the alphabet $\{a,b\}$. **(Nov/Dec 2006)**

Context Free Grammar (CFG)

$R \to R + R$

$R \to a|b|\varepsilon$

11. Write a CFG to generate the set $\{a^m b^n c^p \mid m + n = p \text{ and } p \geq 1\}$. **(Nov/Dec 2006)**

Context Free Grammar (CFG)

$S \to aSc \mid bPc$

$S \to ac \mid bc$

$P \to bc$

12. Show that the grammar $S \to a\ S\ b\ S \mid b\ S\ a\ S \mid \varepsilon$ is ambiguous and what is the language generated by this grammar? **(Nov/Dec 2006)**

Solution :

Given Grammar:

$S \to aSbS \mid bSaS \mid \varepsilon$

Lest most derivations

$\Rightarrow abaSbS (S \to aSbS)$
$\Rightarrow ababS (S \to \varepsilon)$
$\Rightarrow abab\ (S \to \varepsilon)$

(ii) $S \Rightarrow aSbS$

$\Rightarrow abSaSbS \quad (S \to bSaS)$

$\Rightarrow abaSbS \quad (S \to \varepsilon)$

$\Rightarrow ababS \quad (S \to \varepsilon)$

$\Rightarrow abab \quad (S \to \varepsilon)$

The given grammar has two distinct leftmost derivation tress. Hence it is ambignous.

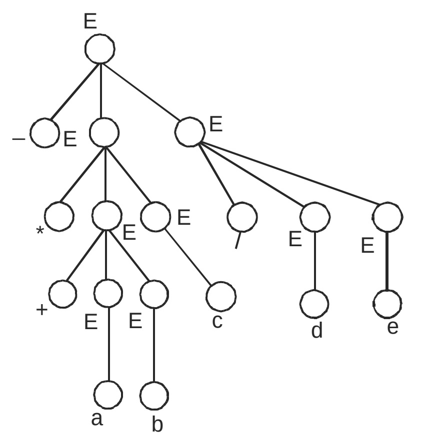
Language $L = \{ w \in (a,b)^* \mid a,b \in w \text{ of even length} \}$

13. Write a grammar to recognize all prefix expressions involving all binary arithmetic operators. Construct parse tree for the sentence "-* + *a b c* / *d e*" using your grammar. **(Nov/Dec 2006)**

Solution :

Grammar with binary arithmetic operators as prefix expressions.

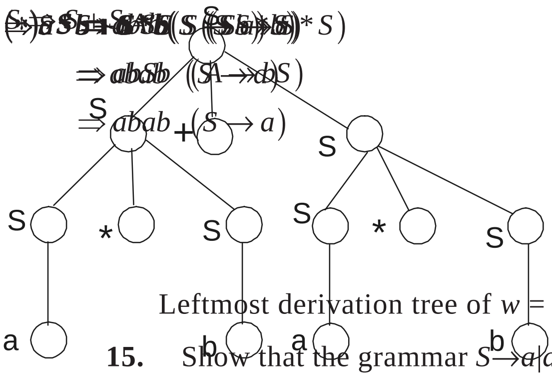
$E \to +EE \mid *EE \mid -EE \mid /EE \mid a \mid b \mid c \mid d \mid e$. Parse tree representation for the given w:



$w = -*+abc/de \rightarrow$ constructed by combining the leaf level nodes from left to right order.

14. Find a derivation tree of $a*b + a*b$ given that $a*b+a*b$ is in L(G) where G is given by $S \rightarrow S+S|S*S, S \rightarrow a|b$. **(May/Jun 2007)**

Solution:



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

$\Rightarrow abab \quad (S \rightarrow a)$

Leftmost derivation tree of $w = a*b + a*b$

15. Show that the grammar $S \rightarrow a|abSb|aAb$, $A \rightarrow bS|aAAb$ is ambiguous. **(May/Jun 2007)**

Solution:

Ambiguity is the existence of two or more leftmost or rightmost derivations. Let $w = abab$

$w = abab$ has two different derivations. Hence the grammar is ambiguous.

16. Consider the alphabet $\Sigma = \{a, b, (,), +, *, ., \in)$. Construct a context free grammar that generates all strings in Σ^* that are regular expressions over the alphabet {a, b}.

(Nov/Dec 2007)

Context Free Grammar (CFG)

$R \to R + R$

$R \to (R)$

$R \to R^*$

$R \to R \,.\, R$

$R \to a \mid b \mid \varepsilon$

17. Show that the grammar $S \to a \mid Sa \mid bSS \mid SSb \mid SbS$ is ambiguous. **(Nov/Dec 2007)**

(Nov/Dec 2008)

Solution :

Given $P = \{S \to a \mid Sa \mid bSS \mid SSb \mid SbS\}$

Ambiguity : A CFG G is said to be ambiguous if there exists some $W \in L(G)$ that has two or more leftmost or rightmost derivation trees.

Leftmost derivations fow $w = baaa$

$S \Rightarrow Sa$	$S \Rightarrow bSS$
$\Rightarrow bSSa \; (S \to bSS)$	$\Rightarrow bSaS \; (S \to Sa)$
$\Rightarrow baSa \; (S \to a)$	$\Rightarrow baaS \; (S \to a)$
$\Rightarrow baaa \; (S \to a)$	$\Rightarrow baaa \; (S \to a)$

Derivation trees:

$\therefore$ the given grammar is ambiguous

18. Find out the context free language. **(May/Jun 2009)**

$S \to aSb \mid aAb$

$A \to bAa$

$A \to ba$

Solution

$S \rightarrow aSb \mid a\mathrm{A}\mathrm{b}$

$A \rightarrow bAa$

$A \rightarrow ba$

$S \Rightarrow aSb$

$\Rightarrow aaAbb$

$\Rightarrow aa\underline{ba}bb$

$S \Rightarrow aAb$

$\Rightarrow abab$

$S \Rightarrow aSb$

$\Rightarrow aaSbb$

$\Rightarrow aaaAbbb$

$\Rightarrow aaa\underline{ba}bbb$

$S \Rightarrow aAb$

$\Rightarrow abAab$

$\Rightarrow ab\underline{ba}ab$

$S \Rightarrow aAb$

$\Rightarrow abAab$

$\Rightarrow ab\underline{ba}ab$

L = {The set of strings over $\Sigma=\{a,b\}$ starting with a and ending with b and substring ba}

19. Construct the CFG for the following languages:

(i) $L(G)=\{a^m b^n \mid m \neq n\ m, n>0\}$ and

(ii) $L(G)=\{a^n\ ba^n \mid n \geq 1\}$. **(May/Jun 2009)**

(i) Given :

$L(G) = \{a^m b^n \mid m \neq n,\ m,\ n > 0\}$

Solution :

CFG :

$S \rightarrow aSb$

$S \rightarrow aC|a|bD|b$

$C \rightarrow aC|a$

$D \rightarrow bD|b$

(ii) Given :

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

Solution :

CFG:

$S \rightarrow aSa$

$S \rightarrow b$

20. (a) Consider the grammar :

(i) $S \rightarrow i\ C\ t\ S$

(ii) $S \rightarrow i\ C\ t\ S\ e\ S$

(iii) $S \rightarrow a$

(iv) $C \rightarrow b$

where i, t, and, e stand for **if, then,** and **else, and C and S for** "conditional" and "statement" respectively.

(1) Construct a leftmost derivation for the sentence w = i b t i b t a e a.

(2) Show the corresponding parse tree for the above sentence.

(3) Is the above grammar ambiguous? If so, prove it.

(4) Remove ambiguity if any and prove that both the grammar produces the same language. **(May/Jun 2010)**

Solution:

w = i b t i b t a e a

Leftmost derivation 1:

$S \Rightarrow i\ C\ t\ S$

$\Rightarrow i\ b\ t\ S\ (C \rightarrow b)$

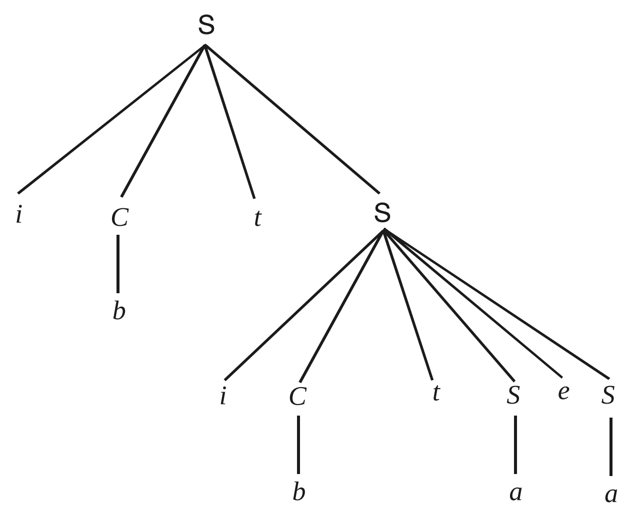
$\Rightarrow i\ b\ t\ i\ C\ t\ SeS\ (S \rightarrow i\ C\ t\ SeS)$

$\Rightarrow i\ b\ t\ i\ bt\ SeS\ (C \rightarrow b)$

$\Rightarrow i\ b\ t\ i\ bt\ aeS\ (S \rightarrow a)$

$\Rightarrow i\ b\ t\ i\ bt\ aea\ (S \rightarrow a)$

(2)



(3) Leftmost derivation 2:

$$\begin{aligned} S &\Rightarrow i\ C\ t\ S\ e\ S \\ &\Rightarrow i\ b\ t\ S\ e\ S\ (C \rightarrow b) \\ &\Rightarrow i\ b\ t\ i\ C\ t\ S\ e\ S\ (S \rightarrow iCtS) \\ &\Rightarrow i\ b\ t\ i\ b\ t\ S\ e\ S\ (C \rightarrow b) \\ &\Rightarrow i\ b\ t\ i\ b\ t\ a\ e\ S\ (S \rightarrow a) \\ &\Rightarrow i\ b\ t\ i\ b\ t\ a\ e\ a\ (S \rightarrow a) \end{aligned}$$

For the string $w = i\ b\ t\ i\ b\ t\ a\ e\ a$, the given grammar has two leftmost derivations. Therefore it is ambiguous.