UNIT IV

CONTEXT- FREE LANGUAGES AND SIMPLIFICATION OF CONTEXT-FREE GRAMMAR

UNIT IV SYLLABUS

- Ambiguity in Context Free Grammar, Language of a Context Free Grammar, Applications of Context Free Grammar, Pumping Lemma for Context Free Grammar, Normal Forms for Context Free Grammar - Chomsky Normal Form, Greibach Normal Form, Context-Free Languages and Derivation Trees, Leftmost and Rightmost derivations, Sentential forms, Construction of Reduced Grammars, Elimination of null and unit productions

Context Free Grammar

- Context free grammar is a formal grammar which is used to generate all possible strings in a given formal language.

- Context free grammar G can be defined by four tuples as:

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

- **N** is a set of non-terminal symbols.
- **T** is a set of terminals where **N ∩ T = NULL.**
- **P** is a set of rules, **P: N → (N U T)***, i.e., the left-hand side of the production rule **P** does have any right context or left context.
- **S** is the start symbol.

Example

- The grammar $(\{A\}, \{a, b, c\}, P, A)$,

  $P : A \to aA, A \to abc.$

- The grammar $(\{S, a, b\}, \{a, b\}, P, S)$,

  $P: S \to aSa, S \to bSb, S \to \varepsilon$

- The grammar $(\{S, F\}, \{0, 1\}, P, S)$,

  $P: S \to 00S \mid 11F, F \to 00F \mid \varepsilon$

Capabilities of CFG

- Context free grammar is useful to describe most of the programming languages.
- If the grammar is properly designed then an efficient parser can be constructed automatically.
- Using the features of associatively & precedence information, suitable grammars for expressions can be constructed.
- Context free grammar is capable of describing nested structures like: balanced parentheses, matching begin-end, corresponding if-then-else's & so on.

Applications of CFG

- Context Free Grammar (CFG) is of great practical importance. It is used for following purposes-
-
- For defining programming languages
- For parsing the program by constructing syntax tree
- For translation of programming languages
- For describing arithmetic expressions
- For construction of compilers

Derivation

- Derivation is a sequence of production rules. It is used to get the input string through these production rules.
- During parsing we have to take two decisions.
- These are as follows:

- We have to decide the non-terminal which is to be replaced.
- We have to decide the production rule by which the non-terminal will be replaced.
- We have two options to decide which non-terminal to be replaced with production rule.

Left-most Derivation

- In the left most derivation, the input is scanned and replaced with the production rule from left to right.

- So in left most derivatives we read the input string from left to right.

Example

- **Production rules:**
- S = S + S
- S = S - S
- S = a | b |c
- **Input:**
- a - b + c

- **The left-most derivation is:**
- S = S + S
- S = S - S + S
- S = a - S + S
- S = a - b + S
- S = a - b + c

Right-most Derivation

- In the right most derivation, the input is scanned and replaced with the production rule from right to left.

- So in right most derivatives we read the input string from right to left.

Example

- **Production rules:**
- S = S + S
- S = S - S
- S = a | b |c
- **Input:**
- a - b + c

- **The right-most derivation is:**
- S = S - S
- S = S - S + S
- S = S - S + c
- S = S - b + c
- S = a - b + c

Parse Tree

- Parse tree is the graphical representation of symbol. The symbol can be terminal or non-terminal.

- In parsing, the string is derived using the start symbol. The root of the parse tree is that start symbol.

- It is the graphical representation of symbol that can be terminals or non-terminals.

- Parse tree follows the precedence of operators.
- The deepest sub-tree traversed first.
- So, the operator in the parent node has less precedence over the operator in the sub-tree.

The parse tree follows these points:

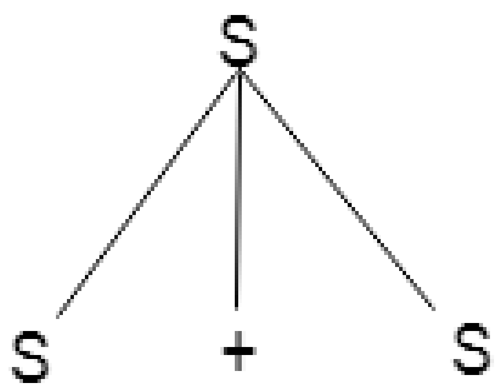
- All leaf nodes have to be terminals.
- All interior nodes have to be non-terminals.
- In-order traversal gives original input string.

Example

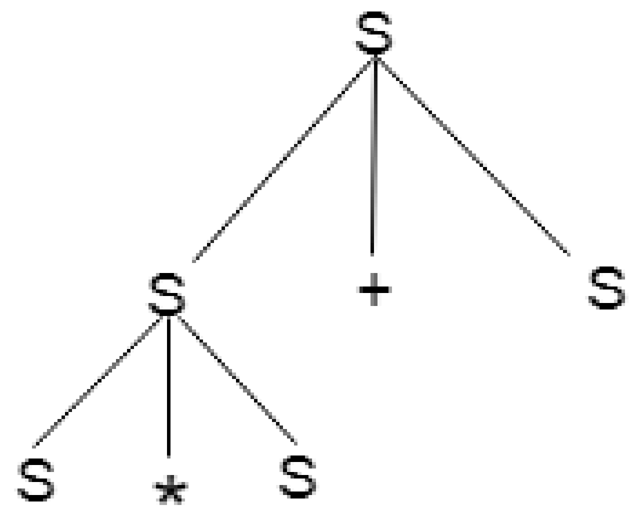
- **Production rules:**
- S= S + S | S * S
- S = a|b|c

- **Input:**
- a * b + c

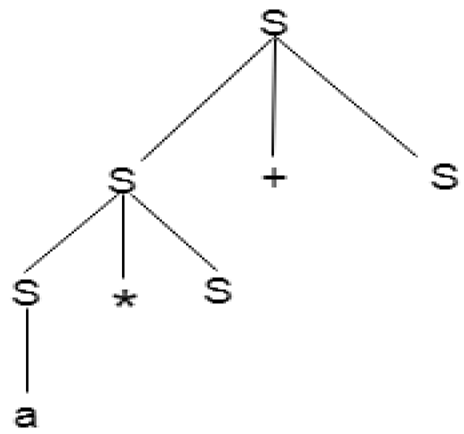
Step 1



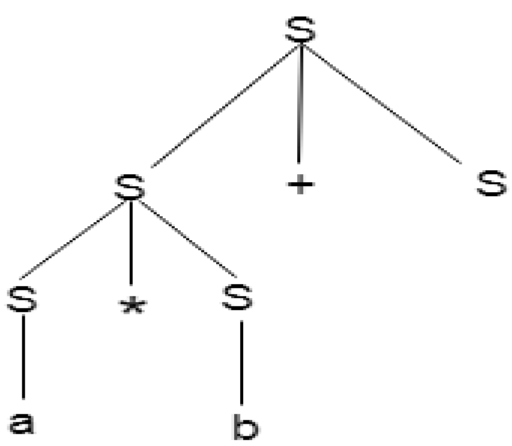
Step 2



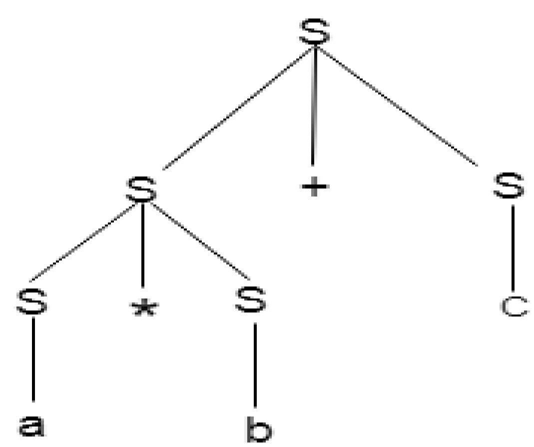
Step 3



Step 4



Step 5



POLLING QUESTIONS

1. Which of the following statement is false?

   a) Context free language is the subset of context sensitive language
   b) Regular language is the subset of context sensitive language
   c) Recursively ennumerable language is the super set of regular language
   d) Context sensitive language is a subset of context free language

2. Which of the following statement is correct?

a) All Regular grammar are context free but not vice versa
b) All context free grammar are regular grammar but not vice versa
c) Regular grammar and context free grammar are the same entity
d) None of the mentioned

Significance of CFG

- Context free languages strike a balance between what is easy enough for a computer to understand and what is expressive enough for a human to use.

- Mathematical expressions as well as large chunks of human languages can be modeled by context free grammars.

- Therefore they are the basis of most programming languages and human-readable data formats.

Ambiguity in CFG

- A grammar is said to be **ambiguous** if there exists more than one leftmost derivation or more than one rightmost derivative or more than one parse tree for the given input string.

- If the grammar is not ambiguous then it is called **unambiguous.**

- If a context free grammar **G** has more than one derivation tree for some string **w ∈ L(G)**, it is called an **ambiguous grammar**.

- There exist multiple right-most or left-most derivations for some string generated from that grammar.

Example 1

- **Problem:** Check whether the grammar G with production rules –

$$X \rightarrow X+X \mid X*X \mid X \mid a$$

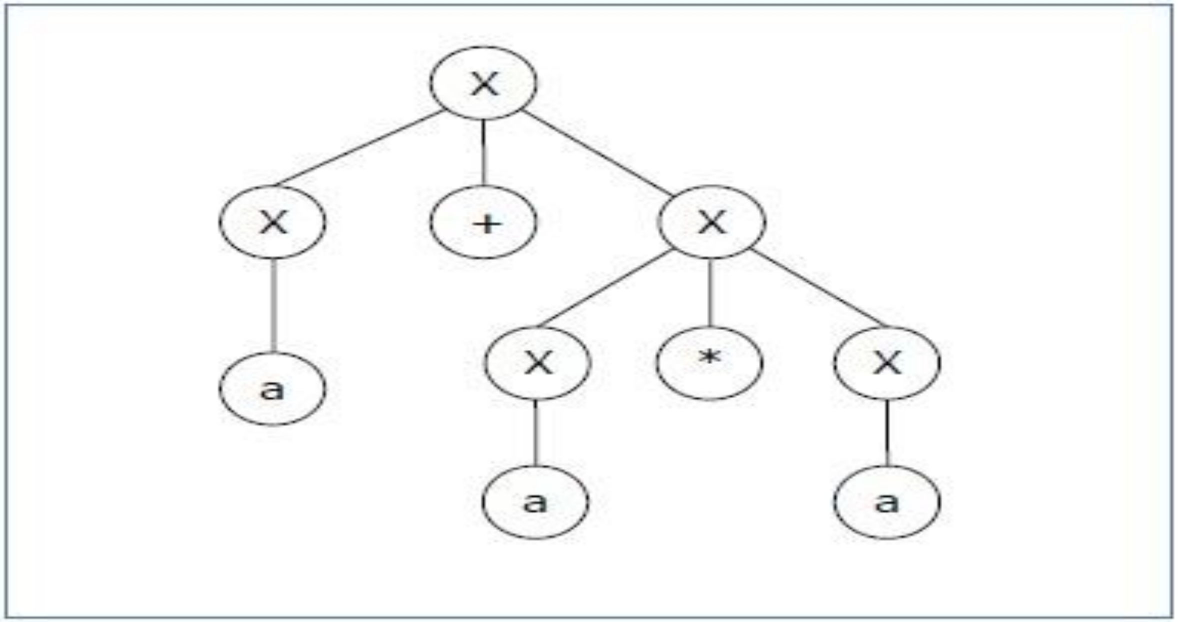
is ambiguous or not.

- **Solution**

- Let's find out the derivation tree for the string **"a+a*a".** It has two leftmost derivations.
- **Derivation 1 –**

X → X+X → a +X → a+ X*X → a+a*X → a+a*a

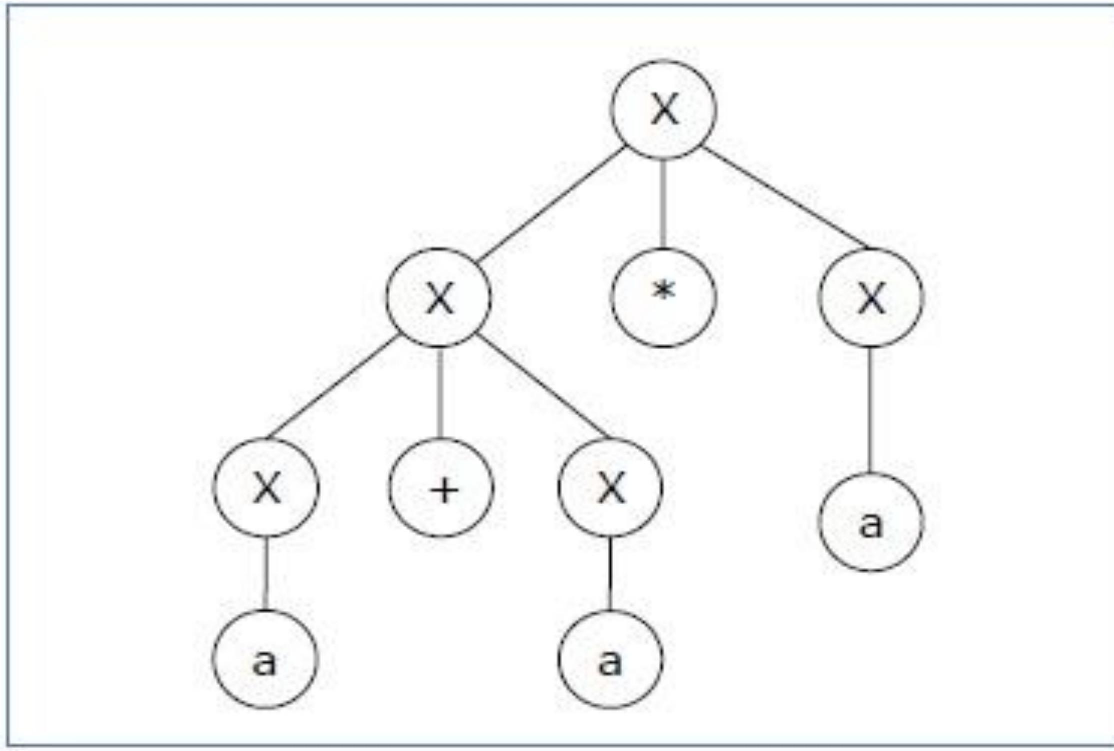
- **Parse Tree 1 –**



- **Derivation 2 –**

X → X*X → X+X*X → a+ X*X → a+a*X → a+a*a

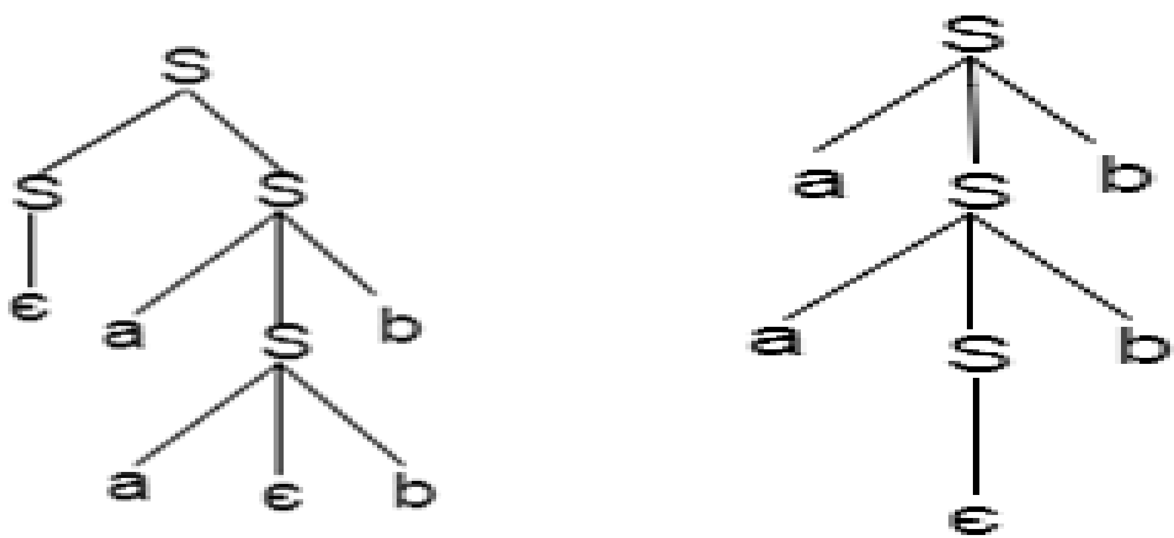
- **Parse Tree 2 –**



Since there are two parse trees for a single string "a+a*a", the grammar **G** is ambiguous.

Example 2

- P: S = aSb | SS |∈
- For the string **aabb**, the above grammar generates two parse trees:



Practice Questions

1. Check whether the given grammar is ambiguous or not-

   $S \rightarrow SS \mid a \mid b$

2. Check whether the given grammar is ambiguous or not-

   $S \rightarrow A / B,\ A \rightarrow aAb / ab,\ B \rightarrow abB / \in$

3. Check whether the given grammar is ambiguous or not-

   $S \rightarrow AB / C,\ A \rightarrow aAb / ab,\ B \rightarrow cBd / cd,\ C \rightarrow aCd / aDd,$
   $D \rightarrow bDc / bc$

4. Check whether the given grammar is ambiguous or not-

$$R \rightarrow R + R \;/\; R \,.\, R \;/\; R^* \;/\; a \;/\; b$$

5. Check whether the given grammar is ambiguous or not-

$$S \rightarrow aSbS \;/\; bSaS \;/\; \in$$

CFL Closure Property

- Context-free languages are **closed** under –
- Union
- Concatenation
- Kleene Star operation

Union

- Let L_1 and L_2 be two context free languages. Then **$L_1 \cup L_2$ is also context free.**
- **Example**
- Let $L_1 = \{ a^n b^n , n > 0\}$. Corresponding grammar G_1 will have P: $S1 \rightarrow aAb|ab$
- Let $L_2 = \{ c^m d^m , m \geq 0\}$. Corresponding grammar G_2 will have P: $S2 \rightarrow cBb|\ \varepsilon$
- Union of L_1 and L_2, $L = L_1 \cup L_2 = \{ a^n b^n \} \cup \{ c^m d^m \}$
- The corresponding grammar G will have the additional production $S \rightarrow S1 \mid S2$

Concatenation

- If L_1 and L_2 are context free languages, then L_1L_2 is also context free.
- Example
- Union of the languages L_1 and L_2,
- $L = L_1L_2 = \{ a^nb^nc^md^m \}$
- The corresponding grammar G will have the additional production $S \rightarrow S1\ S2$

LOVELY PROFESSIONAL UNIVERSITY

Kleene Star

- If L is a context free language, then **L*** is also context free.
- Example
- Let $L = \{ a^n b^n , n \geq 0\}$. Corresponding grammar G will have P: $S \rightarrow aAb \mid \varepsilon$
- Kleene Star $L_1 = \{ a^n b^n \}^*$
- The corresponding grammar G_1 will have additional productions $S1 \rightarrow S\ S1 \mid \varepsilon$

- Context-free languages are **not closed** under –
- **Intersection –** If L1 and L2 are context free languages, then L1 ∩ L2 is not necessarily context free.
- **Intersection with Regular Language –** If L1 is a regular language and L2 is a context free language, then L1 ∩ L2 is a context free language.
- **Complement –** If L1 is a context free language, then L1’ may not be context free.

CFG Simplification

- In a CFG, it may happen that all the production rules and symbols are not needed for the derivation of strings.
- Besides, there may be some null productions and unit productions.
- Elimination of these productions and symbols is called **simplification of CFGs.**

- Simplification essentially comprises of the following steps –

- Reduction of CFG
- Removal of Unit Productions
- Removal of Null Productions

Reduction of CFG

- CFGs are reduced in two phases –
- **Phase 1** – Derivation of an equivalent grammar, **G'**, from the CFG, **G**, such that each variable derives some terminal string.

- **Derivation Procedure** –
- Step 1 – Include all symbols, $\mathbf{W_1}$, that derive some terminal and initialize **i=1**.
- Step 2 – Include all symbols, $\mathbf{W_{i+1}}$, that derive $\mathbf{W_i}$.
- Step 3 – Increment **i** and repeat Step 2, until $\mathbf{W_{i+1} = W_i}$.
- Step 4 – Include all production rules that have $\mathbf{W_i}$ in it.

- **Phase 2 –** Derivation of an equivalent grammar, **G”**, from the CFG, **G’**, such that each symbol appears in a sentential form.

- **Derivation Procedure** –
- Step 1 – Include the start symbol in $\mathbf{Y_1}$ and initialize $\mathbf{i = 1}$.
- Step 2 – Include all symbols, $\mathbf{Y_{i+1}}$, that can be derived from $\mathbf{Y_i}$ and include all production rules that have been applied.
- Step 3 – Increment **i** and repeat Step 2, until $\mathbf{Y_{i+1} = Y_i}$.

Problem

1. Find a reduced grammar equivalent to the grammar G, having production rules,

   P: $S \rightarrow AC \mid B$, $A \rightarrow a$, $C \rightarrow c \mid BC$, $E \rightarrow aA \mid e$

Removal of Unit Productions

- Any production rule in the form A → B where A, B ∈ Non-terminal is called **unit production**..

- Removal Procedure –
- Step 1 – To remove **A → B**, add production **A → x** to the grammar rule whenever **B → x** occurs in the grammar. [x ∈ Terminal, x can be Null]
- Step 2 – Delete **A → B** from the grammar.
- Step 3 – Repeat from step 1 until all unit productions are removed.

Problem

1.Remove unit production from the following –

$S \rightarrow XY, X \rightarrow a, Y \rightarrow Z \mid b, Z \rightarrow M, M \rightarrow N, N \rightarrow a$

Removal of Null Productions

- In a CFG, a non-terminal symbol **'A'** is a nullable variable if there is a production **A → ε** or there is a derivation that starts at **A** and finally ends up with

  ε: A → .... → ε

- Removal Procedure
- **Step 1** – Find out nullable non-terminal variables which derive ε.
- **Step 2** – For each production **A → a**, construct all productions **A → x** where **x** is obtained from **'a'** by removing one or multiple non-terminals from Step 1.
- **Step 3** – Combine the original productions with the result of step 2 and remove **ε - productions**.

Problem

1. Remove null production from the following –

$S \rightarrow ASA \mid aB \mid b,\ A \rightarrow B,\ B \rightarrow b \mid \in$

2. Remove null production from the following –

$S \rightarrow ABAC, A \rightarrow aA \mid \in, B \rightarrow bB \mid \in, C \rightarrow c$

POLLING QUESTIONS

1. **Context free language are closed under**

A. union, intersection

B. union, kleene closure

C. intersection, complement

D. complement, kleene closure

2. If G = ({S}, {a}, {S -> SS), S),

then language generated by G is

A. L (G) = ϕ

B. L(G) = a^n

C. L (G) = a^*

D. L (G) = $a^n b a^n$

3. A given grammar is called ambiguous if

A. two or more productions have the same non-terminal on the left hand side

B. a derivation tree has more than one associated sentence

C. there is a sentence with more than one derivation tree corresponding to it

D. brackets are not present in the grammar

4. **Which of the following derivations does a top-down parser use while parsing an input string? The input is assumed to be scanned in left to right order**

(A) Leftmost derivation

(B) Leftmost derivation traced out in reverse

(C) Rightmost derivation

(D) Rightmost derivation traced out in reverse

5. **Which among the following is the root of the parse tree?**

(A) Production P

(B) Nonterminal V

(C) Terminal T

(D) Starting symbol S

Chomsky Normal Form

- A CFG is in Chomsky Normal Form if the Productions are in the following forms –

  $A \rightarrow a$

  $A \rightarrow BC$

  $S \rightarrow \varepsilon$

- where A, B, and C are non-terminals and **a** is terminal.

Algorithm to Convert into Chomsky Normal Form

- **Step 1 –** If the start symbol **S** occurs on some right side, create a new start symbol **S’** and a new production **S’→ S**.
- **Step 2 –** Remove Null productions. (Using the Null production removal algorithm discussed earlier)
- **Step 3 –** Remove unit productions. (Using the Unit production removal algorithm discussed earlier)

- **Step 4 –** Replace each production **A** → **B_1…B_n** where **n > 2** with **A** → **B_1C** where **C** → **B_2 …B_n**. Repeat this step for all productions having two or more symbols in the right side.

- **Step 5 –** If the right side of any production is in the form **A** → **aB** where a is a terminal and **A, B** are non-terminal, then the production is replaced by **A** → **XB** and **X** → **a**. Repeat this step for every production which is in the form **A** → **aB**.

Problem

1. Convert the following CFG into CNF

   $S \rightarrow ASA \mid aB,\ A \rightarrow B \mid S,\ B \rightarrow b \mid \varepsilon$

2. Convert the following CFG into CNF

   $S \rightarrow a \mid aA \mid B, A \rightarrow aBB \mid \varepsilon, B \rightarrow Aa \mid b$

3. Convert the following CFG into CNF

   $S \rightarrow ASB\ A \rightarrow aAS|a|\varepsilon\ B \rightarrow SbS|A|bb$

POLLING QUESTIONS

1. Suppose A->xBz and B->y, then the simplified grammar would be:

   a) A->xyz
   b) A->xBz|xyz
   c) A->xBz|B|y
   d) none of the mentioned

2. Given grammar G:

S->aS|A|C

A->a

B->aa

C->aCb

Find the set of variables thet can produce strings only with the set of terminals.

a) {C}

b) {A,B}

c) {A,B,S}

d) None of the mentioned

3. Given grammar:

S->aS|A

A->a

B->aa

Find the number of variables reachable from the Starting Variable?

a) 0

b) 1

c) 2

d) None of the mentioned

4. Given a Grammar G:

S->aA

A->a

A->B

B->A

B->bb

Which among the following will be the simplified grammar?

a) S->aA|aB, A->a, B->bb

b) S->aA|aB, A->B, B->bb

c) S->aA|aB, A->a, B->A

d) None of the mentioned

Greibach Normal Form

- A CFG is in Greibach Normal Form if the Productions are in the following forms –

  $A \rightarrow b$

  $A \rightarrow bD_1 \ldots D_n$

  $S \rightarrow \varepsilon$

  where A, $D_1, \ldots, D_n$ are non-terminals and b is a terminal.

Algorithm to Convert a CFG into Greibach Normal Form

- **Step 1** – If the start symbol **S** occurs on some right side, create a new start symbol **S’** and a new production **S’ → S**.
- **Step 2** – Remove Null productions. (Using the Null production removal algorithm discussed earlier)
- **Step 3** – Remove unit productions. (Using the Unit production removal algorithm discussed earlier)
- **Step 4** – Remove all direct and indirect left-recursion.
- **Step 5** – Do proper substitutions of productions to convert it into the proper form of GNF.

Problem

1. Convert the following CFG into GNF

   $S \rightarrow XY \mid Xn \mid p,\ X \rightarrow mX \mid m,\ Y \rightarrow Xn \mid o$

2. Convert the following CFG into GNF

   $S \rightarrow XB \mid AA, A \rightarrow a \mid SA, B \rightarrow b, X \rightarrow a$

3. Convert the following CFG into GNF

   $S \rightarrow CA \mid BB, B \rightarrow b \mid SB, C \rightarrow b, A \rightarrow a$

POLLING QUESTIONS

1. Which of the following does not have left recursions?

   a) Chomsky Normal Form
   b) Greibach Normal Form
   c) Backus Normal Form
   d) All of the mentioned

2. Which of the following grammars are in Chomsky Normal Form:

a) S->AB|BC|CD, A->0, B->1, C->2, D->3
b) S->AB, S->BCA|0|1|2|3
c) S->ABa, A->aab, B->Ac
d) All of the mentioned

3. The format: A->aB refers to which of the following?

a) Chomsky Normal Form
b) Greibach Normal Form
c) Backus Normal Form
d) None of the mentioned

4. Every grammar in Chomsky Normal Form is:

a) regular
b) context sensitive
c) context free
d) all of the mentioned

Pumping Lemma for CFG

- Lemma
- If **L** is a context-free language, there is a pumping length **p** such that any string **w ∈ L** of length ≥ **p** can be written as **w = uvxyz**, where **vy ≠ ε**, **|vxy| ≤ p**, and for all **i ≥ 0, $uv^ixy^iz \in L$**.

- If L is a CFL, there exists an integer n, such that for all $x \in L$ with $|x| \geq n$, there exists u, v, x, y,z $\in \Sigma *$, such that x = uvxyz, and

(1) $|vxy| \leq n$
(2) $|vy| \geq 1$
(3) for all $i \geq 0$: $uv^ixy^iz \in L$

Applications of Pumping Lemma

- Pumping lemma is used to check whether a grammar is context free or not.

Problem

1. Find out whether the language **L = {$a^n b^n c^n$ | n ≥ 1}** is context free or not.

2. Show that L={ww|w is {0,1}* is not context free.

POLLING QUESTIONS

1. In pumping lemma for context free language

a) We start by assuming the given language is context free and then we get contradict

b) We first convert the given language into regular language and then apply steps on

c) Both (a) and (b)

d) None of these

2. The Greibach normal form grammar for the language L = {an bn+1 | n ≥ 0 } is

- **a.**S → aSB, B →bB I λ
- **b.**S → aSB, B →bB I b
- **c.**S → aSB I b, B→b
- **d.**S → aSB I b

3. Consider the following grammar:

A->e

B->aAbC

B->bAbA

A->bB

The number of productions added on the removal of the nullable in the given grammar:

a) 3

b) 4

c) 2

d) 0

4. Given grammar G:

S->aS| AB

A-> e

B-> e

D-> b

Reduce the grammar, removing all the e productions:

a) S->aS| AB| A| B, D-> b

b) S->aS| AB| A| B| a, D-> b

c) S->aS| AB| A| B

d) None of the mentioned

5. Given grammar G:

(1)S->AS

(2)S->AAS

(3)A->SA

(4)A->aa

Which of the following productions denies the format of Chomsky Normal Form?

a) 2,4

b) 1,3

c) 1, 2, 3, 4

d) 2, 3, 4