Unit 4: Context Free Grammars and Languages

We turn our attention away from regular languages to context free languages. These languages have a neutral, recursive notation called Context-free Grammars.

Context-free Grammars have played a central role in compiler technology. We introduce the context-free grammar notation and show how grammars define languages.
- There is an automaton like notation called the push down automaton that describes context free languages.

Components of the Grammar

1) There is a finite set of variables, also called nonterminals or syntatic categories. Each variable represents a language i.e. set of strings.
2) There is a finite set of symbols that form the strings of the language being defined called as Terminals or Terminal symbols.
3) There is a finite set of productions or rules that represent the recursive definition of a language. Each production consists of:
   - A variable that is being defined by the production. This variable is often called as head of the production. (production head)
   - The production symbol →
   - A string of zero or more terminals and variables. This string called the body of the production represents one way to form strings in the language of the variable of the head. (Eg: S → aA → production Body)

   We leave the terminals unchanged whereas the variables (non-terminals) can be replaced by terminals or non-terminals
4) One of the variable represents the language being defined, it is called as the start symbol. The left hand side of production symbol → in the first production is called as start symbol which is denoted by letter S

* Each production starts with a variable
* Followed by ↑ combinations of variables or terminals.
* Followed by

Context Free Grammars:

Grammars:

Definition: A Grammar G is a 4-tuple or quadruple

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

- V is set of variables or non-terminals
- T is set of terminals
- P is set of productions. Each production is of the form $\alpha \to \beta$ where α is a string in $(V \cup T)^+$ and hence α can not be $\in$ and $\in$ can not occur on the ~~right~~ left hand side of any production but β is a string in $(V \cup T)^*$ and hence it includes $\in$ also. i.e. head of the production cannot be $\in$ [i.e null string]
- S is the start symbol.

The various notations used are shown below:

→ The following are the terminals: [Alphabet Σ]
- The keywords such as if, for, while, do-while etc
- Digits from 0-9
- Symbols such as +, -, *, / etc.
- The lower case letters near the beginning of the alphabets such as a, b, c, d etc.
- The bold faced letters such as id

→ The following are non-terminals: [All capital letters in production]
- The lower case names such as expression, operator, operand, statements etc.
- The capital letters near the beginning of the alphabets such as A, B, C, D etc.
- The letter S represents the start symbol.

→ The lower case letters near the end of the alphabets such as (u, v) w, x, y, z represent string of terminals.

→ The capital letters near the end of the alphabet such as X, Y, Z etc represents grammar symbols. The grammar symbol can be terminal or non-terminal.

→ Lower-case greek letters such as α and β are strings consisting of terminals and/or non-terminals.

Chomsky Hierarchy:

Naom Chomsky who is the founder of formal languages theory has classified the grammar into various categories.

Q. What are various types of Grammars / Explain Chomsky Hierarchy / Explain the classification of Grammars.

The Grammar can be classified as:

Type 0 Grammar (Phrase Structured Grammar)
Type 1 Grammar (Context Sensitive Grammar)
Type 2 Grammar (Context Free Grammar)
Type 3 Grammar (Regular Grammar)

Type 0 Grammar:

Definition → Type 0 Grammar is 4-tuple $G=(V,T,P,S)$ where
V is set of variables or non-terminals
T is set of Terminals
S is start symbol
P is set of productions of the form $\alpha \to \beta$ where

A Grammar $G=(V,T,P,S)$ is said to be type 0 grammar or unrestricted grammar or phrase structured grammar if all the productions are of the form $\alpha \to \beta$ where

Explanation for the above

$\alpha \in (V \cup T)^+$ and $\beta \in (V \cup T)^*$

- In this type of grammar there are no restrictions on length of α and β. The only restriction is that α cannot be ϵ but β can be ϵ.
- This is the largest family of grammars more powerful than all other types of grammars. Any language can be obtained from this grammars.
- The language generated from this grammar is called type 0 language or recursively enumerable language.
- Only turing machine (TM) can recognize this language.

$S \to aAb \mid \epsilon$
$aA \to bAb$
$bA \to a$

Here, $V=\{S,A\}$
$T=\{a,b\}$

b, aAb, bAbb

Type 1 Grammar or Context Sensitive Grammar:

Context Sensitive Grammar is a 4-tuple, $G=(V,T,P,S)$ where
V is set of variables or non-terminals
T is set of terminals
P is set of productions of the form $\alpha \to \beta$ where
$\alpha, \beta \in (V \cup T)^+$ and $|\beta| \geq |\alpha|$
S is the start symbol.

Note: There is restriction on the length β. The length of β must be atleast as much as length of α.

Eg:
$S \to aAb$
$aA \to bA$
$bA \to aa$

→ ϵ cannot not appear on the left side or the right hand side of the production.
→ Its an ϵ-free grammar
- Language generated from this grammar is called type 1 language or context sensitive language
- Linear Bounded Automata (LBA) can be constructed to recognize the language generated from this grammar.

Type 2 Grammar or Context Free Grammar:

Context Free Grammar (CFG) is a 4-tuple or quadruple $G=(V,T,P,S)$ where
V is set of variables or non-terminals
T is set of terminals
P is set of productions of the form $A \to \alpha$ where
$A \in V$ (i.e A is a variable) and $\alpha \in (V \cup T)^*$
S is the start symbol
- The symbol ϵ can appear on the right hand of production
- The language Generated from this grammar is type 2 language or Context free language.

Language of CFG: If $G_1=(V,T,P,S)$ is a CFG, the language of G denoted by $L(G_1)$ is the set of terminal strings that have derivations from the start symbol. that is

$L(G) = \{w \text{ in } T^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w\}$. If a language L is the language of some context free Grammar, then L is said to be Context free language (CFL).

- Push down Automaton (PDA) can be constructed to recognize the language generated from this grammar

Eg: $S \to aA \mid bB \mid \epsilon$

$A \to bA \mid a$

$B \to aB \mid b \mid \epsilon$

Type 3 Grammar or Regular Grammars

The Grammar $G = (V, T, P, S)$ is a type 3 grammar or regular Grammar if and only if the grammar is right linear or left linear.

→ A Grammar $G = (V, T, P, S)$ is said to be right linear if all the productions are of the form

$A \to wB$

or

$A \to w$ where A, B are variables and $w \in T^*$ (string of zero or more terminals)

[i.e if each production body has atmost one variable and that variable is at the right end]

→ A Grammar $G = (V, T, P, S)$ is left linear if all the productions are of the form

$A \to Bw$

or

$A \to w$

where $A, B \in V$ and $w \in T^*$

i.e if each production body has atmost one variable and that variable is at the left end.

Eg: The following grammar

$S \to aAA \mid bbB \mid \epsilon$

$A \to aA \mid b$

$B \to bB \mid a \mid \epsilon$

is right linear Grammar.

- Here ϵ is present on R.H.S including strings of terminals.
- if non-terminal (variable) is present on R.H.S of any production then only one non terminal should be present and it has to be the right most symbol on R.H.S

The Grammar,

$S \to Aaa \mid Bbb \mid \epsilon$

$A \to Aa \mid b$

$B \to Bb \mid a \mid \epsilon$

is left linear Grammar. Note that ϵ and strings of terminals can appear on R.H.S of any production and if non-terminal is present on R.H.S of any production, only one non-terminal should be present and it has to be the left most symbol on R.H.S.

Consider the grammar

$S \to aA$

$A \to aB \mid b$

$B \to Ab \mid a$

In this grammar, each production is either left linear or right linear. But the grammar is not either left linear or right linear. Such type of grammar is called linear grammar [i.e grammar with atmost one non-terminal (variable) where the non-terminal can be leftmost or rightmost is called the linear grammar]

→ The language generated from this grammar is regular language.

→ Finite Automaton can be constructed to recognize the grammar language generated from this grammar.

Ex: 1) $S \to aAb \mid \epsilon$ 2) $S \to aAb$ 3) $S \to aAb \mid \epsilon$

$aA \to bA$ $aA \to bA \mid b$ $aA \to bA$

$bA \to a$ $bA \to a$ $bA \to a$

Type 0 Type 0 Type 0

d) $S \to aAb$
$aA \to bA$
$bA \to a$

Type 0

e) $S \to aAb$
$aA \to bA$
$bA \to aa$

Type 0, Type 1

f) $S \to aA/bB/\epsilon$
$A \to b/\epsilon$
$B \to a$

Type 0, Type 2, Type 3 (Right)

g) $S \to AbBb$
$A \to B$
$B \to C$
$C \to \epsilon$

Type 0, type 2, type 3 grammar. (Left)

(Explain this before)

② Grammar from finite Automata:

Not explained → since it is very difficult for all.

→ How to obtain Grammar from Finite Automaton very easily.

The procedure is:

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a FA accepting Lange L where

$Q = \{q_0, q_1, \dots q_n\}$

$\Sigma = \{a_1, a_2, \dots a_m\}$

A Grammar $G = (V, T, P, S)$ can be constructed where

- $V = \{q_0, q_1, \dots q_n\}$ i.e the states of DFA will be the variables in the grammar.
- $T = \Sigma$ i.e the input alphabets of DFA are the terminals in the grammar
- $S = q_0$ i.e the start state of DFA is the start symbol in the grammar
- The production P from the transitions can be obtained as shown below:

Step 1: If $\delta(q_i, a) = q_j$ then introduce the transition:

$q_i \to a q_j$

Step 2: If $q \in F$, i.e if q is the final state in FA, then introduce the production

$q \to \epsilon$

Using above 2 steps we can convert any finite automaton (DFA or NFA or ϵ-NFA) into grammar.

① Derivations using Grammar:

There are 2 approaches to this: (productions)

- a) The more conventional approach is to use the rules from body to head. That is, we take strings known to be in the language of each of the variables of the body, concatenate them, in the proper order, with any terminals appearing in the body. and infer that resulting string is in the language of the variable in the head. This procedure is called recursive inference

- There is another approach to defining the language of a grammar, in which we use the procedure from head to body. We expand the start symbol using one of its productions (i.e using a production whose head is the start symbol). We further expand the resulting string by replacing one of the variables by the body of one of its productions and so on until we derive a strings consisting entirely of terminals. The language of the grammar is all strings of terminals that we can obtain in this way. This use of grammars is called derivation.

* Questions on Grammar from Fihite Automata

(Follow method 1 since it is easy)

d) i) Obtain grammars to generate string consisting of any no. of a's.

Soln: DFA to accept strings consisting of any no. of a's is given by

Method 2: start → (S) with a self-loop a

Method 1: $\epsilon, a, aa, aaa \dots$

Min string accepted is ϵ

$S \to \epsilon$ & $S \to \epsilon / aS$

transitions for DFA is given by:

S is the final state as well as the start state.

Transition:

$\delta[S,a) = S$

$\Rightarrow S \to aS$

Since S is the final state we have

$S \to \epsilon$

∴ The Grammar for the string consisting of any no. of a's is

$S \to \epsilon$
$S \to aS$ i.e $\boxed{S \to \epsilon | aS}$

The language generated by the grammar can be written as

$L = \{a^n : n \geq 0\}$

2) Obtain grammar to generate string consisting of atleast one a

Soln: For atleast one a, strings accepted are

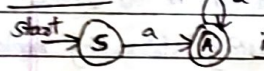
a, aa, aaa, ...

Other than ϵ, min. string is a.

method 1 (Direct looking at possible strings)

$S \to a$, $S \to aS$
$S \to aS$ } $S \to a | aS$

method 2:



i.e $\delta(S,a) = A$ $S \to aA$

$\delta(A,a) = A$ $A \to aA$

Since A is final state $A \to \epsilon$

i.e

$\boxed{\begin{array}{l} S \to aA \\ A \to aA | \epsilon \end{array}}$ → OR

The language for the grammar is

$L = \{a^n : n \geq 1\}$

3) Note: 1) From Method 1 and Method 2 one or more different grammars may exist that generate same language.

2) For each final state introduce ϵ-transition. For example, in a DFA, if the states A and B are final states, then in the grammar we should introduce two ϵ-productions

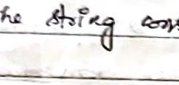
3) Obtain grammar to generate string consisting of any number of a's and b's.

Soln: Any no. of a's and b's means

ϵ, a, b, aa, bb, ab, aab, baa, ...

$S \to \epsilon$
$S \to aS$
$S \to bS$ } $\boxed{S \to \epsilon | aS | bS}$

Method 2:



$\delta(S,a) = S$ $S \to aS$

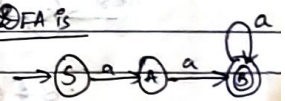
$\delta(S,b) = S$ $S \to bS$

S is final state $S \to \epsilon$.

4) Grammar to generate string consisting of at least two a's.

→ Strings possible are

aa, aaa, aaaa .... DFA is



Minimum string is aa

$S \to aa$
$S \to aS$ } → $\boxed{S \to aa | aS}$

5) Obtain Grammar to generate string consisting of even number of a's

→ Solution is we need to have strings like

$L = \{\epsilon, aa, aaaa, aaaaaa, \text{ 8 a's, 10 a's } ....\}$ or

$L = \{w : n_a(w) \bmod 2 = 0, w \in \{a\}^*\}$

odd no. of a's

$S \to \epsilon$
$S \to aaS$ } $\boxed{S \to \epsilon | aaS}$

DFA is

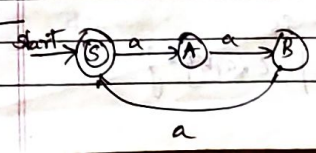
6) Grammar to generate string consisting of multiples of three a's. $[L = \{w | n_a(w) \bmod 3 = 0, w \in \{a\}^*\}$

Strings possible are

$L = \{\epsilon, aaa, aaaaaa, ...\}$

$S \to \epsilon$
$S \to aaaS$ } $\boxed{S \to \epsilon | aaaS}$

DFA



7) Obtain grammar to generate strings of a's and b's with atleast one a (consisting any no/.. of)

→ Language is $L = \{ w : n_a(w) \geq 1 \mid w \in \{a,b\}^* \}$ $(a+b)^m\ a\ (a+b)^n$

Strings accepted are aab is not accepted

a, ab, ba, aa, aba, baa, . . .

$S \to a$
$S \to aS$
$S \to bS$

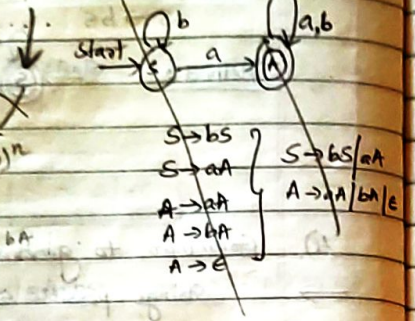
$S \to a \mid aS \mid bS$ ✗

$(a+b)^m\ a\ (a+b)^n$

$S \to AaA$
$A \to \epsilon \mid aA \mid bA$

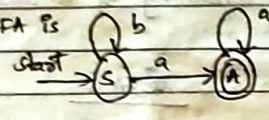
Using FA,

DFA



$S \to bS \mid aA$
$A \to aA \mid bA \mid \epsilon$

DFA is



$\delta(S,b) = S$ $\quad S \to bS$
$\delta(S,a) = A$ $\quad S \to aA$
$\delta(A,a) = A$ $\quad A \to aA$
$\delta(A,b) = A$ $\quad A \to bA$
$A \to \epsilon$

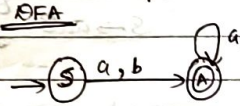
$\Rightarrow$ $S \to bS \mid aA$, $A \to aA \mid bA \mid \epsilon$

Since A is final state,

8) Obtain grammar to generate strings consisting of any no/.. of a's and b's with atleast one a or atleast one b.

Soln: a, b, ab, abb, baa, . . . $L = \{ (a+b)^m (a+b)(a+b)^n \mid m, n \geq 0 \}$

DFA



$S \to a \mid b$
$S \to aS$ bs
$S \to bS$ bbs

$S \to aA \mid bA$
$A \to aA \mid bA \mid \epsilon$

9) Obtain grammar to generate strings of a's and b's whose string length is a multiple of three.

Strings

Soln: Language for the question is

$L = \{ w : |w| \bmod 3 = 0 \mid w \in \{a,b\}^* \}$

Strings accepted are: ϵ, aaa, bbb, abb, bab, aba, ...

DFA is

$S \to \epsilon$
$S \to AAAS$
$A \to a \mid b$

$\Rightarrow$ $S \to \epsilon \mid AAAS$, $A \to a \mid b$

10) Obtain grammar to accept the language $L = \{ b : |w| \bmod 3 > 0$ where $w \in \{a\}^* \}$

Soln: $|w| \bmod 3 > 0$ means remainder can be 1 or 2

Minimum strings accepted are a, aa

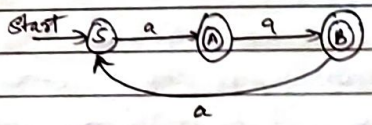
Possible strings are:

$L = \{ a, aa, aaaa, aaaaa, aaaaaaa, \ldots \}$

$S \to a \mid aa$
$S \to aaaS$

$\Rightarrow$ $S \to a \mid aa \mid aaaS$

Using DFA,



The transitions are $\delta(S,a) = A \Rightarrow S \to aA$
$\delta(A,a) = B \quad A \to aB$
$\delta(B,a) = S \quad B \to aS$
Since A, B are final states $A \to \epsilon$
$B \to \epsilon$

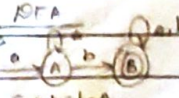
$\Rightarrow$ $S \to aA$, $A \to aB \mid \epsilon$, $B \to aS \mid \epsilon$

Grammars from Regular Expressions

11) Obtain grammars to generate strings of a's and b's having substring ab.

→ The regular expression can be written as $(a+b)^* ab (a+b)^*$

DFA



$S \to bS \mid aA$
$A \to aA \mid bB$
$B \to \epsilon \mid aB \mid bB$

Possible strings are:

$L = \{ab, aab, bab, aaab \ldots\}$

(a+b)* ab (a+b)*

A ab A

$S \to A\ ab\ A$

$A \to \epsilon \mid aA \mid bA$ (i.e A can have ϵ, or any no.. of a's & b's)

12) Grammar to generate strings of a's and b's ending with string ab.

(a+b)* ab

$S \to A\ ab$

$A \to \epsilon \mid aA \mid bA$

13) Begining with ab

ab(a+b)*

$S \to ab\ A$

$A \to \epsilon \mid aA \mid bA$

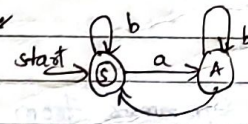
14) Obtain grammar to generate the following language:

$L = \{w : n_a(w) \bmod 2 = 0$ where $w \in \{a,b\}^*\}$

Even no.. of a's is given by

Any no.. of b's is given by

$S \to \epsilon \mid bS$



Strings possible are:

ϵ, aa, b, aab, baa, aba ----

$b^*ab^*ab^*$

$S \to SaSaS$

$S \to \epsilon \mid bS \mid SaSaS$

$S \to bS$

$S \to aA$

$S \to \epsilon$

$A \to bA$

$A \to aS$

$S \to \epsilon \mid bS \mid aA$

$A \to aS \mid bA$

Derivation:

Let $A \to \alpha B \gamma$ and $B \to \beta$ are productions in grammar G, where α, β and γ are strings of terminals and/or non-terminals, A and B are non-terminals. The non-terminal A produces the string $\alpha\beta\gamma$ by replacing the non-terminal B with string β by applying the production $B \to \beta$. which can be written as

$A \Rightarrow \alpha\beta\gamma$

Explain derivation from standard book.

This process of obtaining strings of terminals and/or non-terminals from the start symbol by applying some or all the productions is called derivation.

If a string is obtained by applying only one production then it is called one-step derivation and is denoted by symbol '$\Rightarrow$'. If one or more productions are applied to get the string $\alpha\beta\gamma$ from A, then we write as

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

If zero or more productions are applied to get the string $\alpha\beta\gamma$ from A, then we write

$A \overset{*}{\Rightarrow} \alpha\beta\gamma$

Eg: Consider the grammar shown below from which any arithmetic expression can be obtained

$E \to E+E$

$E \to E-E$

$E \to E*E$

$E \to E/E$

$E \to id$

The non-terminal E is used instead of the expression.

Obtain the string id+id*id and show derivation for the same.

$E \Rightarrow E+E$
$\Rightarrow id+E$
$\Rightarrow id+E*E$
$\Rightarrow id+id*E$
$\Rightarrow id+id*id$

Since the string id + id * id is obtained from start symbol E by applying more than one production, this can be written as

$E \overset{+}{\Rightarrow} id+id*id$.

Sentence or Sentential Form:

Let $G=(V,T,P,S)$ be a grammar. The string w obtained from the grammar G such that $S \overset{+}{\Rightarrow} w$ is called sentence of grammar G.
(Here w is string of terminals)

In the above derivation, id+id*id is the sentence of grammar. If there is a derivation $S \overset{*}{\Rightarrow} \alpha$ where $\alpha \in L(V,T)^*$ contains strings of terminals and/or non-terminals then α is called sentential form of G.
In the above derivation,
E+E, id+E, id+E*E, id+id*E, id+id*id are all sentential forms of the grammar.

Language: A language generated by the Grammar G of grammar can be formally defined as

Let $G=(V,T,P,S)$ be a grammar. The language $L(G)$ generated by the grammar is the set of terminal strings that have derivations from the start symbol which is given as $L(G)=\{w \mid S \overset{*}{\underset{G}{\Rightarrow}} w \text{ and } w \text{ is in } T^*\}$

—The various strings that are elements of L(G) are called sentences.

Consider the grammar

$S \to aCa$
$C \to aCa \mid b$

What is the language generated by this grammar.

$S \Rightarrow aCa$
$\Rightarrow aba$

$S \Rightarrow aCa$
$\Rightarrow aaCaa \Rightarrow aabaa$

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

Grammars for other languages:

1) Obtain grammar to generate the following language.
$L=\{a^n b^n : n \geq 0\}$
(Meaning is equal no. of a's is followed by equal no. of b's

Soln: Minimum string is ∈. (0 a's 0 b's)
Strings accepted are ∈, ab, aabb, aaabbb, ...
$S \to \epsilon$
$S \to aSb$

2) Grammar to generate the following language
$L=\{a^n b^n : n \geq 1\}$
Minimum string is ab
Strings accepted are
ab, aabb, aaabbb, ...

$S \to ab$
$S \to aSb$
} $S \to ab \mid aSb$

3) $L=\{a^{n+1} b^n : n \geq 0\}$
Strings are: a, aab, aaabb, aaaabbb, ...

$S \to a$
$S \to aSb$
} $S \to a \mid aSb$

*

$L=\{0^{n+2}1^n \mid n \geq 1\}$
0001, 00 0011, 000 00111, ...
Soln: $S \to 0A$
$A \to 00$
$B \to 0B1 \mid 01$
(OR)
$S \to 00B1 \mid 000A1$
$A \to \epsilon \mid 0A1$

4) Grammar to generate the language:

$L = \{a^n b^{n+2} : n \geq 0\}$ [No. of a's is followed by 2 more b's than no. of a's]

bb, abbb, aabbbb,

Minimum string is bb

$S \to bb$ } $S \to bb | aSb$ or $S \to AB$

$S \to aSb$ } $A \to \epsilon | aAb$

$B \to bb$

5) Grammar to generate the following language:

$L = \{a^n b^n : n \geq 0\}$. [i.e For every 'a', two b's are to be generated"

Soln: Minimum string is ϵ.

Strings accepted are:

ϵ, abb, aabbbb, aaabbbbbb

$S \to \epsilon$

$S \to aSbb$

6) Obtain grammar G generating set of all palindromes over $\Sigma = \{a, b\}$ — $W = W^R$

Soln: For the given alphabet $\Sigma = \{a, b\}$, the definition of palindrome along with corresponding productions is:

1) ϵ is a palindrome. The equivalent production is $S \to \epsilon$

2) 'a' and 'b' are palindromes. The production is $S \to a|b$

3) The ~~string~~ if w is a palindrome then string aSa and the string bwb are palindromes. The equivalent production is

$\therefore L = \{\epsilon, a, b, aaa, aba, bab, \ldots\}$

$S \to aSa | bSb$

$\therefore$ The grammar to generate strings of palindromes over Σ is given by $G = (V, T, P, S)$

Where $V = \{S\}$

$T = \{a, b\}$

$P = \{ S \to \epsilon$

$S \to a|b$

$S \to aSa | bSb \}$

S is the start symbol.

7) Obtain grammar to generate the following language.

Even Pal $L = \{ww^R$ where $w \in \{a,b\}^*\}$

Soln: Strings that are accepted by the language are:

$L = \{\epsilon, aa, bb, abba, baab, aaaa, bbbb, abbbbba\}$

Here the strings are palindrome of even length. So the (Previous) (above) grammar can be modified and given as:

$S \to \epsilon$

$S \to aSa | bSb.$

Note: In the above grammar if the production $S \to \epsilon$ is replaced by $S \to c$ the resulting grammar still generate the language $L = \{wcw^R | w \in \{a,b\}^*\}$

8) Obtain grammar to generate a language consisting of all the non-palindromes over $\{a, b\}$.

$S \to aSa | bSa$

$S \to aAb | bAa$

$A \to aA | bA | \epsilon$

$L = \{0^m 1^m 2^n \mid m \geq 1 \text{ and } n \geq 0\}$

Soln: Meaning is Equal no/.. of 0's is followed by equal no/.. of 1's which is followed by any no/.. of 2's. With atleast an 01

$$L = \{\underbrace{0^m 1^m}_{A}\ \underbrace{2^n}_{B} \mid m \geq 1 \text{ and } n \geq 0\}$$

$S \to AB$

$A \to 01 \mid 0A1$ (Equal no/. of 0's followed by equal no/.. of 1's)

$B \to \epsilon \mid 2B$ (Any no/.. of 2's)

$S \to A \mid S2$

$A \to 01 \mid 0A1$

10) Obtain grammar to generate the language.

After 10, solve $L = \{w \mid n_a(w) = n_b(w)\}$

Soln: No/. of a's in the string is equal to no/.. of b's.

Strings possible here

$L = \{\epsilon, ab, ba, aabb, abba, abab, baab, baba, bbaa \ldots\}$

There are following productions

i) ϵ has equal a's & b's.

$S \to \epsilon$

2) Equal a's can be followed by Equal b's.

$S \to aSb$

3) Equal b's can be followed by a's

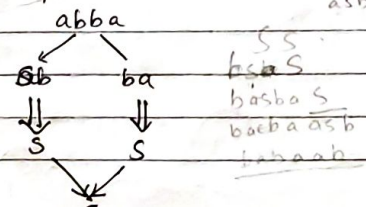
$S \to bSa$.

} Recurring strings.

Above three cases will accept all strings except the strings which begins and ends with the same symbol i.e abba, baab.

To accept this, the production can be

$S \to SS$.



The productions are:

$S \to \epsilon$
$S \to aSb \mid bSa$
$S \to SS$

} [OR]

$S \to aSbS \mid bSaS \mid \epsilon$

11) $L = \{a^{n+2} b^m \mid n \geq 0 \text{ and } m > n\}$

Soln: Strings in the language are:

$L = \{aab, aabb, aabbb, aaabb, aaabbb, aaabbbb\}$

$aabb^*$, $aaabbb^*$ (minimum string)

Equal a's followed by equal b's with atleast an ab

It looks like this is beginning with a and ending with any no/.. of b's. Production can be given in combined form as

$a\ ab\ b^*$

$S \to a\ A\ B$

$A \to ab \mid aAb$

$B \to \epsilon \mid bB$

$S \to AB$

$A \to \epsilon \mid aS$

$B \to b \mid bS$

12) Obtain grammar to generate the language.

$L = \{a^n b^m \mid n \geq 0, m > n\}$

Strings of the language are

$L = \{b, bb, bbb, bbbb, bbbbb, abb, abbb, abbbb,$

$n=0, m \geq 1$ → ϵbb^* ; $n=1, m \geq 2$ → $abbb^*$

$aabbb, aabbbb, aabbbbb \ldots\}$

$n=2, m \geq 3$

$aabbb^*$

$S \to aSb \mid B$

$B \to bB \mid b$

[Strings of a's and b's having b between equal no/.. of a's and b's and ending with any no/.. of b's]

$L = \{\epsilon bb^*, abbb^*, aabbbb^*, aaabbbbb^*\}$

$\epsilon, ab, aabb$; bb^*

A ; bB

$S \to AbB$

$A \to \epsilon \mid aAb$

$B \to \epsilon \mid bB$

[Equal no/. of a's followed by b's which followed by atleast one b] for a's for b's.

OR

$S \to AB$

$A \to \epsilon \mid aA$

$B \to b \mid bB$

$S \to AB$

$A \to aAb \mid \varepsilon$

$B \to b \mid bB$

Text Book.

$S \to AB$

$A \to \epsilon \mid aAb$

$B \to b \mid bB$

13) $L=\{a^n b^{n-3} \mid n\geq 3\}$ a's followed by b's with atleast 3 a's.

Set of Strings generated by the language are:

$L=\{aaa, aaaab, aaaabb, aaaaabbb \ldots\}$

It is observed that aaa is followed by equal no. of a's and equal no. of b's.

$S\to AB$
$A\to aaa$
$B\to aBb|\epsilon$

or

$S\to aaaA$
$A\to aAb|\epsilon$

or $S\to aSb|aaa$

[other way Equal no. of a's followed by aaa followed by equal no. of b's with min one aaa]

14) $L=\{0^i 1^j \mid i\neq j, i\geq 0 \text{ and } j\geq 0\}$ 0's followed by 1's such that Equal no. of 0's and 1's should not be there.

$S\to OS1$ → Equal no. of 0's followed by equal no. of 1's
$S\to A|B$
$A\to 0A|0$ not equal to condition
$B\to 1B|1$

15) What is the language generated by the grammar.

$S\to 0A|\epsilon$
$A\to 1S$

Soln: Strings generated from the grammar are

$L=\{\epsilon, 01, 0101, 010101, \ldots\}$

∴ The language generated is

$L=\{(01)^n \mid n\geq 0\}$

16) Obtain grammar to generate the grammar language:

$L=L_1L_2$

where

$L_1=\{a^n b^m \mid n\geq 0, m>n\}$

$L_2=\{0^n 1^{2n} \mid n\geq 0\}$

Grammar for language L_1 is given by following production

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

where $V=\{S_1, A, B\}$
$T=\{a,b\}$

instead of this ← you can reduce
$S\to aSb|B$
$B\to bB|b$

$S_1\to AB$
$A\to \epsilon|aAb$ ← P=
$B\to b|bB$

$S=\{S_1\}$ is the start symbol.

Grammar for language L_2 is given by following production.

$S_2\to \epsilon|0S_2 11$

Resulting Language $L=L_1L_2$ can be obtained by concatenating the two grammar as

$S\to S_1S_2$
$S_1\to AB$
$A\to \epsilon|aAb$
$B\to b|bB$
$S_2\to \epsilon|0S_2 11$

16) Obtain grammar to generate the language

$L=L_1\cup L_2$

where $L_1=\{a^n b^m \mid n\geq 0, m>n\}$

$L_2=\{0^n 1^{2n} \mid n\geq 0\}$

Soln: The language Grammar for language $L=L_1\cup L_2$ can be obtained as follows:

$S\to S_1|S_2$

i.e It is obtained by the union of S_1, S_2. ie from the start symbol S either S_1 is derived or S_2 is derived

The resulting productions for the grammar are:

$S \to S_1 | S_2$

$S_1 \to AB$

$S_2 \to 0S_2 11 | \epsilon$

$A \to aAb | \epsilon$

$B \to bB | b$

17) Obtain a grammar to generate the language

$L = \{ w \mid |w| \bmod 3 \neq |w| \bmod 2 \}$ on $\Sigma = \{a\}$

0,1 2 3 4 5 6,3 8 9 10 11 12,8

aa, aaa, aaba, aaaaa, aaaaaaaa,

Soln $S \to aa | aaa | aaaa | aaaaa | aaaaaaS$

[you can draw DFA and do it]

(OR)

$S \to aaA | aaaA | aaaaA | aaaaaA$

$A \to aaaaaaA | \epsilon$

18) Grammar to generate the language

$L = \{ w : |w| \bmod 3 \geq |w| \bmod 2 \}$ on $\Sigma = \{a\}$

Soln: $\epsilon, a, aa, aaa, aaaa, aaaaa, aaaaaa, aaaaaaa, aaaaaaaa,$

1, 2, 3, 4, 5, 6, 7, 9

$S \to \epsilon | a | aa | aaaa | aaaaa | aaaaaaS$

↓ if you replace S by others you will get 7 a's, 8 a's, 10 a's, 11 a's ...

11-10

19) Obtain grammar to generate set of all strings with exactly one a when $\Sigma = \{a, b\}$

[Language is $L = \{ w \mid n_a(w) = 1, w \in \{a,b\}^* \}$

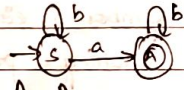
Soln: Exactly one a can have following strings in the language

$L = \{ a, ab, ba, abb, bab, bbbba, abbbbb, bba \ldots \}$

$S \to bS | aA$

$A \to \epsilon | bA$

(OR)



$S \to AaA$

$A \to bB | \epsilon$

20) Obtain grammar to generate set of all strings with atleast one a when $\Sigma = \{a, b\}$

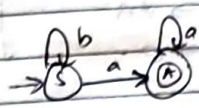
11-10

Soln: Language is $L = \{ w \mid n_a(w) \geq 1, w \in \{a,b\}^* \}$

Strings in the language are:

$L = \{ a, ab, ba, aa, aaa, aa^* bab, \ldots \}$

$S \to aS | a | bS$



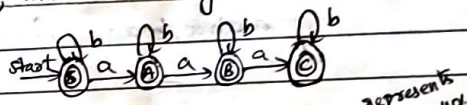
$S \to bS | aA$

$A \to aA | bA | \epsilon$

21) Obtain grammar to generate the set of all strings with no more than three a's. when $\Sigma = \{a, b\}$.

Soln: Language is $L = \{ w \mid n_a(w) \leq 3, w \in \{a,b\}^* \}$.

There can be atmost 3 a's - i.e 0 a's [no a's at all] only one a, only two a's, three a's only.



$S \to bS | aA | \epsilon$ • No a's at all gives $S \to B$ (B represents any no. of b's)

$A \to bA | aB | \epsilon$ • Only one a gives $S \to BaB$

$B \to bB | aC | \epsilon$ • only two a's gives $S \to BaBaB$

$C \to bC | \epsilon$ • only three a's gives $S \to BaBaBaB$

So grammar is:

$S \to BABABAB$

$A \to \epsilon | a$

$B \to \epsilon | bB$

22) Obtain grammar to generate the language

$L = \{ w \mid n_a(w) = n_b(w) + 1 \}$

Meaning is no. of a's in the string is one more than the no. of b's

For Equal no. a's and equal no. of b's we have the grammar. To modify the grammar for the given language we must have one 'a' more than equal no. of a's & b's.

An 'a' with any no. of equal a's and equal b's can be given as

$A \to aAb \mid bAa \mid AA \mid \epsilon$

Equal no. of a's & b's is given by

$S \to AaA$

$A \to aAb \mid bAa \mid AA \mid \epsilon$

$S \to AaA$

$A \to \epsilon \mid aAb \mid bAa \mid AA$.

23) Obtain grammar to generate the language

$L = \{ w \mid n_a(w) > n_b(w) \}$

Soln: Strings are

$L = \{ a, aa, aaa, aaaa, aab, aaabb, aaaab, aaaaab, aba, baa, abaa, baaa, aaabb, \pi \}$

$a, aa, aaa, aaaa$ — aa^*

$L = \{ a, aa, aaa, aaaa, aab, aba, baa, aaab, abaa, aaba, baaa, \ldots \}$.

24) Obtain grammar to generate a language of strings of 0's and 1's having a substring 000

Soln: $L = \{ (0+1)^m 000 (0+1)^n \mid m, n \geq 0 \}$

or

$(0+1)^* 000 (0+1)^*$

$S \to A\,000\,A$

$A \to \epsilon \mid 0A \mid 1A$

25) Obtain grammar to generate integers. Derive 1965 from this. Obtain a derivation for unsigned nos. 1965.

Soln: The integers can have +, − or optional (ε) sign.

Production for this is

$S \to + \mid - \mid \epsilon$

A no. can be formed by from any digits 0, 1, 2 ... 9. The production to obtain these digits is

$D \to 0 \mid 1 \mid 2 \mid \ldots \mid 9$

Integer Examples

+3

−5

.5

5256

+5256

−5236

Number N can be a digit or more than one digit.

$N \to D \mid ND \mid DN$

An Integer is either a unsigned number N or a sign (plus or minus) followed by number N. The production for this is

$I \to N \mid SN$

∴ The productions are

$I \to N \mid SN$ (Signed or unsigned nos.)

$N \to D \mid DN \mid ND$

$D \to 0 \mid 1 \mid 2 \mid 3 \mid \ldots \mid 9$

$S \to + \mid - \mid \epsilon$

if $S \to \epsilon$ then $I \to N$ is (not required)

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

$V = \{ I, N, D, S \}$

$T = \{ +, -, 0, 1, 2, \ldots 9 \}$

$S = I$ is the start symbol.

$I \Rightarrow N$ 1965

$\Rightarrow ND$

$\Rightarrow N5$

$\Rightarrow ND5$

$\Rightarrow N65$

$\Rightarrow ND65$

$\Rightarrow N965$

$\Rightarrow D965$

$\Rightarrow 1965$

$I \Rightarrow SN$ +1965

Arithmetic Expression

$E \rightarrow E+E$

$E \rightarrow E-E$

$E \rightarrow E*E$

$E \rightarrow E/E$

$E \rightarrow (E)$

$E \rightarrow a|b$ id

26) Obtain grammar to generate an arithmetic expression using the operators +, -, *, / and ^ (indicating power). An identifier can start with any of the letters from {a, b, c} and can be followed by zero or more symbols from {a, b, c}

Doubt

An arithmetic expression can be defined recursively as.

1) An expression E can be an identifier.

2) The arithmetic expressions are

E+E

E-E

E*E

E/E

E^E

An identifier I of length atleast one can be generated using any combinations of a, b, c and the production can be given as

$I \rightarrow a|b|c|Ia|Ib|Ic$

∴ Productions are

$E \rightarrow E$ | $E \rightarrow I$

$E \rightarrow E+E$ | $I \rightarrow a|b|c|aI|bI|cI$

$E \rightarrow E-E$

$E \rightarrow E*E$

$E \rightarrow E/E$

27) Let $G=(V,T,P,S)$ be a CFG where

Date / /20

$V=\{S\}$

$T=\{a,b\}$

$P=\{S \rightarrow aSa \mid bSb \mid \epsilon\}$

S is the start symbol. What is the language generated by this grammar?

Strings are:

$L=\{\epsilon, aa, bb, aaaa, bbbb, abba, baab, \ldots\}$ the strings of a's and b's with equal ... for ww^R

$L=\{ww^R \mid w \in \{a,b\}^*\}$

28) Obtain grammar to generate the foll. lang.

$L=\{a^n b^m c^k \mid n+2m=k, \text{ for } n\geq 0, m \geq 0\}$

Soln. From the above language, k is expressed in terms of n and m.

∴ The language L can be written as

$L=\{a^n b^m c^{n+2m} \mid n\geq 0, m\geq 0\}$

This is equivalent to

$L=\{a^n b^m c^{2m} c^n \mid n\geq 0, m\geq 0\}$

$a^n c^n$ – Equal no. of a's ended with equal no. of c's.

$b^m c^{2m} \mid m\geq 0$ takes strings as ϵ, bcc, bbcccc, bbbcccccc ...

production for $b^m c^{2m} \mid m \geq 0$ can be written as

$A \rightarrow \epsilon \mid bAcc$

For $a^n c^n$ and $b^m c^{2m}$ combined together, production is

$S \rightarrow aSc \mid A$

$A \rightarrow \epsilon \mid bAcc$

Question Paper Questions

29) $L=\{a^i b^j c^k \mid i=j+k, i\geq 0, j\geq 0, k\geq 0\}$ $S \rightarrow aSb \mid aa/bb$

30) $L=\{a^n w w^R b^n \mid w \in \{0,1\}^* \text{ and } n \geq 2\}$ $A \rightarrow 0A0 \mid 1A1 \mid \epsilon$

31) $L=\{a^k b^m c^n \mid m+n=k, m,n \geq 1\}$ $\rightarrow S \rightarrow aSc \mid aAc$; $A \rightarrow aAb \mid ab$

32) $L=\{a^n b^m c^k \mid k=m+n, n,m,k \geq 0\}$ $\rightarrow S \rightarrow aSc \mid A$

$\rightarrow aSc \mid A$

$\rightarrow \mid aAb$

Derivation:

Left most & Right Most derivation:

Left most Derivation: In the derivation process, if at each step the left most variable is replaced by one of its production bodies, then the derivation is called left most derivation.

Left most derivation is indicated by $\underset{lm}{\Rightarrow}$

Right most Derivation:

Right most derivation is indicated by $\underset{rm}{\Rightarrow}$

Eg: Consider the grammar for arithmetic expression.

$E \to E+E$
$E \to E*E$
$E \to E/E$
$E \to id$

Obtain string id+id*id using left most and right most derivation

Left most

$E \underset{lm}{\Rightarrow} E+E$
$\underset{lm}{\Rightarrow} id+E$
$\underset{lm}{\Rightarrow} id+E*E$
$\underset{lm}{\Rightarrow} id+id*E$
$\underset{lm}{\Rightarrow} id+id*id$
$\therefore E \overset{+}{\underset{lm}{\Rightarrow}} id+id*id$

Right most

$E \underset{rm}{\Rightarrow} E+E$
$\underset{rm}{\Rightarrow} E+E*E$
$\underset{rm}{\Rightarrow} E+E*id$
$\underset{rm}{\Rightarrow} E+id*id$
$\underset{rm}{\Rightarrow} id+id*id$
$\therefore \boxed{E \overset{+}{\underset{rm}{\Rightarrow}} id+id*id}$

Sentential form:

Let $G=(V,T,P,S)$ be a CFG, Any string α in $(V \cup T)^*$ which is derivable from the start symbol S such that $S \overset{*}{\Rightarrow} \alpha$ is called sentential form of G

* If there is derivation of the form $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$ where at each step of the derivation process a left most variable is replaced by one of its production bodies, then α is called left sentential form

Eg: $E \underset{lm}{\Rightarrow} E+E$
$\underset{lm}{\Rightarrow} id+E$
} left sentential form.

* If $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha$, then α is right sentential form.

Eg: $E \underset{rm}{\Rightarrow} E+E$
$\underset{rm}{\Rightarrow} E+E*E$
$\underset{rm}{\Rightarrow} E+E*id$

Q): Obtain the left most derivation for the string aaabbabbba using the following

$S \to aB \mid bA$
$A \to aS \mid bAA \mid a$
$B \to bS \mid aBB \mid b$

$S \underset{lm}{\Rightarrow} aB$
$\underset{lm}{\Rightarrow} aaBB$
$\underset{lm}{\Rightarrow} aaaBBB$
$\underset{lm}{\Rightarrow} aaabBB$
$\underset{lm}{\Rightarrow} aaabbB$
$\underset{lm}{\Rightarrow} aaabbaBB$
$\underset{lm}{\Rightarrow} aaabbabB$
$\underset{lm}{\Rightarrow} aaabbabbA$
$\underset{lm}{\Rightarrow} aaabbabbba$

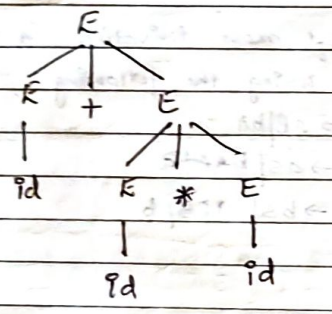
Derivation Tree (Parse Tree)

Each derivation can be represented using tree called parse tree.

Definition: Let $G = (V, T, P, S)$ be a CFG. The parse tree (derivation tree) for G are trees with following conditions:

1) The root node is labeled by a variable in V
2) Each interior node is labelled by a variable in V.
3) Every node (vertex) has a label which is in $(V \cup T \cup \epsilon)$
4) Each leaf node is labeled by either a variable in V, or terminal in T or ϵ. If the leaf is labeled by ϵ, then $\epsilon (\forall v \in \epsilon)$ it must be the only child of its parent.
5) If an interior node is labeled A, and its children are labelled as $X_1, X_2, X_3, \ldots X_n$ from left then $A \to X_1 X_2 X_3 \cdots X_n$ must be a production in P.

Q: Parse tree for the left most derivation to obtain string id + id * id is



If we read the leaf nodes from left to right we obtain string id + id * id which is called as **yield of the tree**

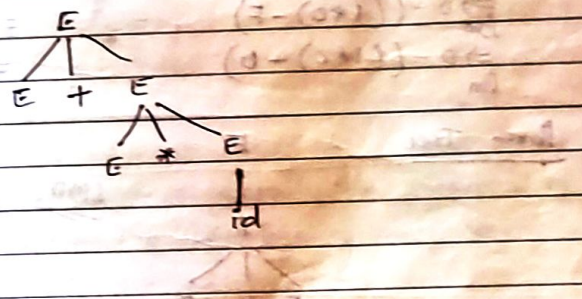
Yield of a Tree : The yield of a tree is the string of symbols obtained by concatenating the leaves of the parse tree from left to right which is derived from the root variable and the yield of tree is always terminal string.

Partial Parse tree or Partial Derivation Tree :

Partial Parse tree is same as Parse except the property 4 which says in partial parse tree Each leaf node is labeled by either a variable in V, Terminal in T or E i.e Every leaf node has label from $(V \cup T \cup \epsilon)$

Eg:

$$E \underset{lm}{\Rightarrow} E+E$$
$$\underset{lm}{\Rightarrow} E+E*E$$
$$\underset{lm}{\Rightarrow} E+E*id$$



5) Consider CFG with productions

$E \to E*T \mid T$
$T \to F-T \mid F$
$F \to (E) \mid 0 \mid 1$

Write the leftmost derivation, right most derivation and parse tree for the string $0-((1*0)-0)$

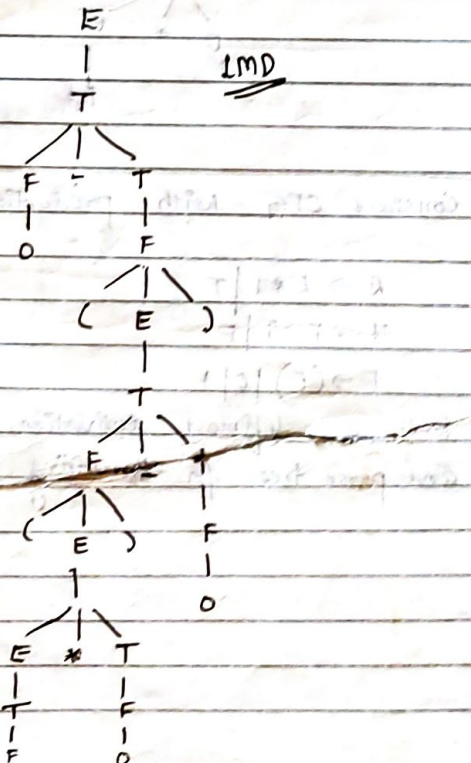
Left Most

$E \Rightarrow T$
$\Rightarrow F-T$
$\Rightarrow 0-T$
$\Rightarrow 0-F$
$\Rightarrow 0-(E)$
$\Rightarrow 0-(T)$
$\Rightarrow 0-(F-T)$
$\Rightarrow 0-((E)-T)$
$\Rightarrow 0-((E*T)-T)$
$\Rightarrow 0-((T*T)-T)$
$\Rightarrow 0-((F*T)-T)$
$\Rightarrow 0-((1*T)-T)$
$\Rightarrow 0-((1*F)-T)$
$\Rightarrow 0-((1*0)-T)$
$\Rightarrow 0-((1*0)-F)$
$\Rightarrow 0-((1*0)-0)$

Right Most

$E \Rightarrow T$
$\Rightarrow F-T$
$\Rightarrow F-F$
$\Rightarrow F-(E)$
$\Rightarrow F-(T)$
$\Rightarrow F-(F-T)$
$\Rightarrow F-(F-F)$
$\Rightarrow F-(F-0)$
$\Rightarrow F-((E)-0)$
$\Rightarrow F-((E*T)-0)$
$\Rightarrow F-((E*F)-0)$
$\Rightarrow F-((E*0)-0)$
$\Rightarrow F-((T*0)-0)$
$\Rightarrow F-((F*0)-0)$
$\Rightarrow F-((1*0)-0)$
$\Rightarrow 0-((1*0)-0)$

Parse Tree



LMD

Yield of

Q2) Write LMD, RMD and parse tree for the string "+*-xyzy" using the grammar

$E \rightarrow +EE \mid *EE \mid -EE \mid x \mid y$

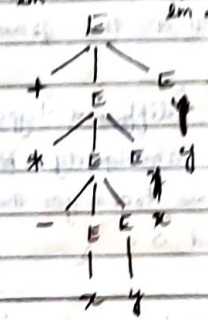
Soln: Left most derivation

$E \Rightarrow +EE$
$\Rightarrow +*EEE$
$\Rightarrow +*-EEEE$
$\Rightarrow +*-xEEE$
$\Rightarrow +*-xyEE$
$\Rightarrow +*-xyzE$
$\Rightarrow +*-xyzy$

Right most derivation

$E \Rightarrow +EE$
$\Rightarrow +Ey$
$\Rightarrow +*EEy$
$\Rightarrow +*E-Eyy$
$\Rightarrow +*E-xyy$
$\Rightarrow +*-EExyy$
$\Rightarrow +*-xEyyy$
$\Rightarrow +*-xyyyy$

Parse Tree:



Ans:

Q3) The following grammars generates the grammars of the language consisting of all strings of even length

$S \rightarrow AS \mid \epsilon$
$A \rightarrow aa \mid ab \mid ba \mid bb$

Give leftmost and Rightmost derivations for the following strings:

a) aabbba
b) baabab
c) aaabbb

Ambiguous Grammars :-

A context free Grammar $G = (V, T, P, S)$ is ambiguous if there is atleast one string w in T^* for which we can find two different parse trees, each with root labeled S and yield w.
(two or more)

→ If each string has atmost one parse tree in the grammar then the grammar is unambiguous.

Note that after applying leftmost derivation or right most derivation even though the derivations look different and if the structure of parse tree obtained are same, then we cannot conclude that the grammar is ambiguous.

- If is not the multiplicity (different types) of derivations for the same string cause ambiguity, But it is the existence of two or more parse trees for the same string w derived from the root labeled S.

*

> Only condition to prove that the grammar is ambiguous is for atleast one string you must have 2 (or more) different parse trees (different in terms of structure of parse tree) either by applying left most derivation or right most derivation

Q) 1) Consider the grammar for arithmetic expression

$E \to E + E$
$E \to E - E$
$E \to E * E$
$E \to E / E$
$E \to (E)$
$E \to id$

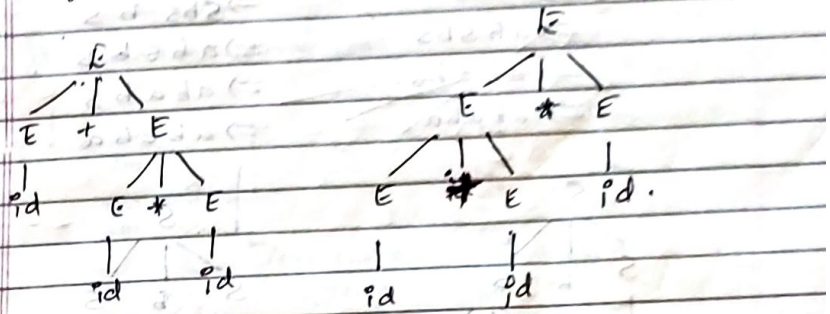
Whether the grammar is ambiguous?

Soln: The string id+id*id can be obtained as

$E \Rightarrow E + E$
$\Rightarrow id + E$
$\Rightarrow id + E * E$
$\Rightarrow id + id * E$
$\Rightarrow id + id * id$

$E \Rightarrow E * E$
$\Rightarrow E + E * E$
$\Rightarrow id + E * E$
$\Rightarrow id + id * E$
$\Rightarrow id + id * id$

Derivations tree (parse tree) for the above two left most derivations can be shown as:



Since the string id+id*id has two different parse trees, the grammar is ambiguous.

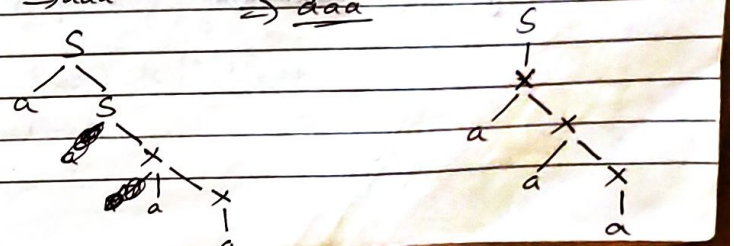
Q2) Is the following grammar ambiguous?

$S \to aS \mid X$
$X \to aX \mid a$

Soln: Consider a string aaa, for this string we can drive parse trees as

$S \Rightarrow aS$
$\Rightarrow aX$
$\Rightarrow aaX$
$\Rightarrow aaa$

$S \Rightarrow X$
$\Rightarrow aX$
$\Rightarrow aaX$
$\Rightarrow aaa$



Since two different parse trees can be drawn for string aaa, the grammar is ambiguous.

Q3) Is the grammar ambiguous.

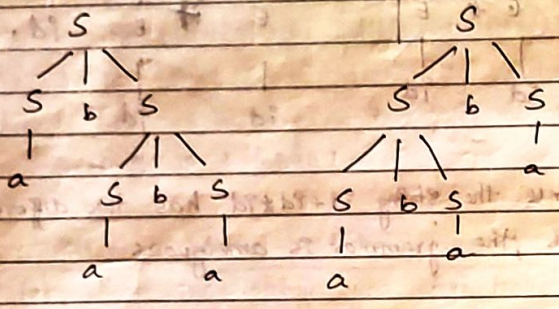
$S \to SbS$

$S \to a$

Soln: Consider the string abab, the different derivations are

$S \Rightarrow SbS$
$\Rightarrow abS$
$\Rightarrow abSbS$
$\Rightarrow ababS$
$\Rightarrow ababa$

$S \Rightarrow SbS$
$\Rightarrow SbSbS$
$\Rightarrow abSbS$
$\Rightarrow ababS$
$\Rightarrow ababa$



Since the string abab has two different parse trees, the grammar is ambiguous.

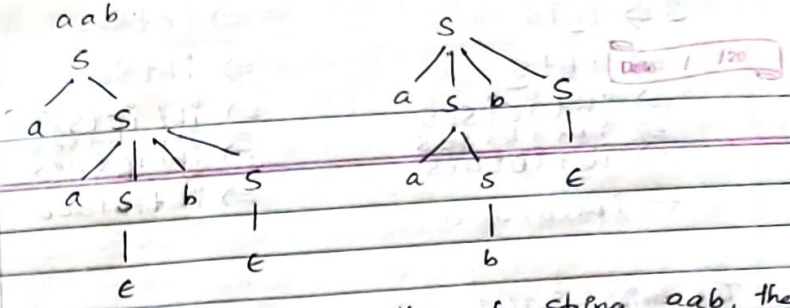
Q4) Consider the grammar

$S \to as \mid aSbS \mid \epsilon$

Is the above grammar ambiguous? Show that the string aab has two

i) parsetrees
ii) Left Most derivations
iii) Right Most derivations.

i) aab



Since two parse trees are there for string aab, the grammar is ambiguous.

ii) 2 Left most derivations.

$S \underset{lm}{\Rightarrow} as$
$\underset{lm}{\Rightarrow} aaSbS$
$\underset{lm}{\Rightarrow} aa\epsilon bS$ (ϵ need not represent)
$\underset{lm}{\Rightarrow} aabS$
$\underset{lm}{\Rightarrow} aab$

$S \underset{lm}{\Rightarrow} aSbS$
$\underset{lm}{\Rightarrow} aaSbS$
$\underset{lm}{\Rightarrow} aabS$
$\underset{lm}{\Rightarrow} aab$

iii) Two RMD's are.

$S \underset{rm}{\Rightarrow} as$
$\underset{rm}{\Rightarrow} aaSbS$
$\underset{rm}{\Rightarrow} aaSb\epsilon$
$\underset{rm}{\Rightarrow} aaSb$
$\underset{rm}{\Rightarrow} aab$

$S \underset{rm}{\Rightarrow} aSbS$
$\underset{rm}{\Rightarrow} aSb$
$\underset{rm}{\Rightarrow} aaSb$
$\underset{rm}{\Rightarrow} aab$

Q5) Is the following grammar ambiguous??

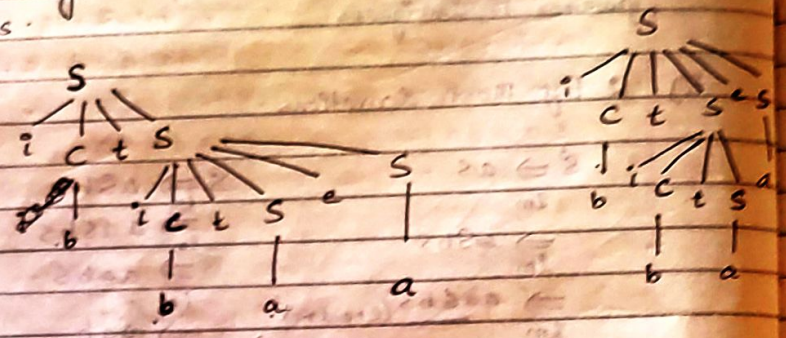
$S \to icts \mid ictSeS \mid a$

$C \to b$

$S \Rightarrow i\,c\,t\,S$
$\Rightarrow ibtS$
$\Rightarrow ibt\,i\,c\,t\,S\,e\,S$
$\Rightarrow ibtibtaes$
$\Rightarrow ibtibtaea$

$S \Rightarrow i\,c\,t\,S\,e\,S$
$\Rightarrow ibtS\,e\,S$
$\Rightarrow ibt\,i\,c\,t\,S\,e\,S$
$\Rightarrow ibtibtaes$
$\Rightarrow ibtibtaea$

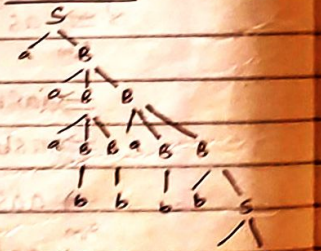
The string ibtictaea can has two different parse trees.



Q6) Obtain the string aaabbabbba by applying left most derivation and the parse tree for the grammar shown below. Is it possible to obtain the same string again by applying leftmost derivation but by selecting different productions?

$S \rightarrow aB \mid bA$
$A \rightarrow aS \mid bAA \mid a$
$B \rightarrow bS \mid aBB \mid b$

Parse tree



Soln:
$S \Rightarrow aB$
$\Rightarrow aaBB$
$\Rightarrow aaaBBB$
$\Rightarrow aaabBB$
$\Rightarrow aaabbaBB$
$\Rightarrow aaabbabB$
$\Rightarrow aaabbabbS$
$\Rightarrow aaabbabbbA$
$\Rightarrow aaabbabbba$

$S \Rightarrow aB$
$\Rightarrow aaBB$
$\Rightarrow aaaBBB$
$\Rightarrow aaabBB$
$\Rightarrow aaabbaB$
$\Rightarrow aaabbabB$
$\Rightarrow aaabbabbS$
$\Rightarrow aaabbabbbA$
$\Rightarrow aaabbabbba$

The left most derivation of string to obtain string aaabbabbba

$S \Rightarrow aB$
$\Rightarrow aaBB$
$\Rightarrow aaaBBB$
$\Rightarrow aaabBB$
$\Rightarrow aaabbAB$
$\Rightarrow aaabbaSB$
$\Rightarrow aaabbabAB$
$\Rightarrow aaabbabaBB$
$\Rightarrow aaabbabbBA$
$\Rightarrow aaabbabbba$

$S \Rightarrow ab$
$\Rightarrow aaBb$
$\Rightarrow aaaBBb$
$\Rightarrow aaabBBb$
$\Rightarrow aaabbABb$
$\Rightarrow aaabbaBBb$
$\Rightarrow aaabbabBb$
$\Rightarrow aaabbabbAb$
$\Rightarrow aaabbabbba$

Inherently ambiguous grammars:

① A context-free language L is said to be inherently ambiguous if all its grammars are ambiguous. If even one grammar for L is unambiguous, then L is an unambiguous language.

② The grammars from which the language is derived is called inherently ambiguous grammar.

For eg: $E \rightarrow E+E \mid E-E$
$E \rightarrow E*E$
$E \rightarrow E/E$
$E \rightarrow (E)$
$E \rightarrow id$

} Grammar for arithmetic expression.

has been proved as ambiguous grammars.

We can obtain unambiguous grammar for same lang

$E \rightarrow T*E \mid T/E \mid T$
$T \rightarrow T+F \mid T-F \mid F$
$F \rightarrow (E) \mid I$
$I \rightarrow id$

(OR)

$E \rightarrow E+T \mid E-T \mid T$
$T \rightarrow T*F \mid T/F \mid F$
$F \rightarrow (E) \mid I$
$I \rightarrow id$