Ambiguity in context free grammars:

In computer science, an ambiguous grammar is a context-free grammar for which there exists a string that can have more than one leftmost derivation, while an unambiguous grammar is a context-free grammar for which every valid string has a unique leftmost derivation. Many languages admit both ambiguous and unambiguous grammars, while some languages admit only ambiguous grammars. Any non-empty language admits an ambiguous grammar by taking an unambiguous grammar and introducing a duplicate rule or synonym (the only language without ambiguous grammars is the empty language). A language that only admits ambiguous grammars is called an inherently ambiguous language, and there are inherently ambiguous context-free languages. Deterministic context-free grammars are always unambiguous, and are an important subclass of unambiguous CFGs; there are non-deterministic unambiguous CFGs, however.

For real-world programming languages, the reference CFG is often ambiguous, due to issues such as the dangling else problem. If present, these ambiguities are generally resolved by adding precedence rules or other context-sensitive parsing rules, so the overall phrase grammar is unambiguous.

If a context free grammar G has more than one derivation tree for some string w $\in L(G)$, it is called an ambiguous grammar. There exist multiple right-most or left-most derivations for some string generated from that grammar.

Problem:

Check whether the grammar G with production rules, X → X+X | X*X |X| a is ambiguous or not.

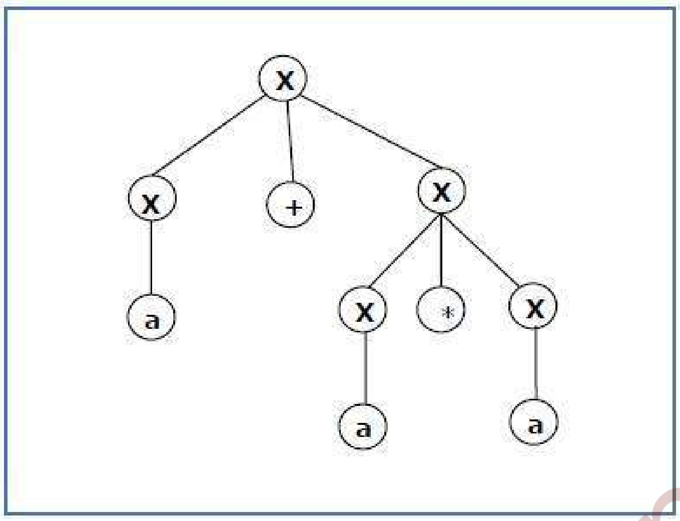
Solution:

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

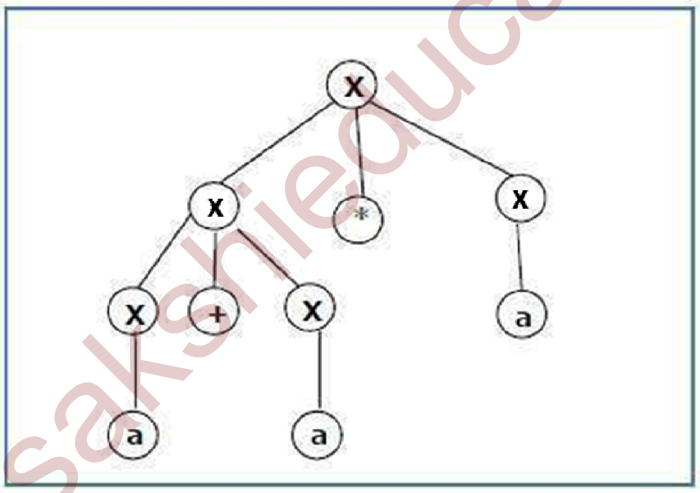
Parse tree 1:



Derivation 2 −

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

Parse tree 2:



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

Trivial language

The simplest example is the following ambiguous grammar for the trivial language, which consists of only the empty string:

A → A | ε

…meaning that a production can either be itself again, or the empty string. Thus the empty string has leftmost derivations of length 1, 2, 3, and indeed of any length, depending on how many times the rule A → A is used.

This language also has the unambiguous grammar, consisting of a single production rule:

$A \rightarrow \varepsilon$

…meaning that the unique production can only produce the empty string, which is the unique string in the language.

In the same way, any grammar for a non-empty language can be made ambiguous by adding duplicates.

Unary string

The regular language of unary strings of a given character, say 'a' (the regular expression a*), has the unambiguous grammar:

$A \rightarrow aA \mid \varepsilon$

…but also has the ambiguous grammar:

$A \rightarrow aA \mid Aa \mid \varepsilon$

These correspond to producing a right-associative tree (for the unambiguous grammar) or allowing both left- and right- association.

Addition and Subtraction

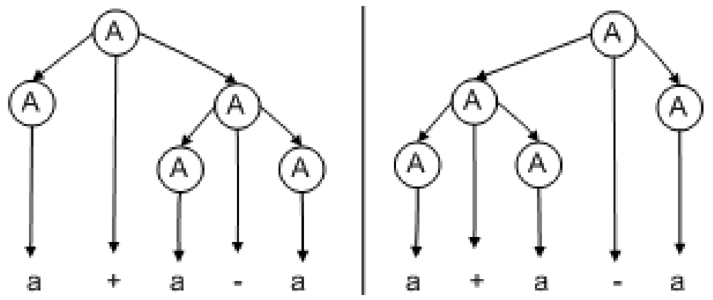
The context free grammar

$A \rightarrow A + A \mid A - A \mid a$

is ambiguous since there are two leftmost derivations for the string a + a + a:

$A \rightarrow A + A$	$A \rightarrow A + A$
$\rightarrow a + A$	$\rightarrow A + A + A$ (First A is replaced by A+A. Replacement of the second A would yield a similar derivation)
$\rightarrow a + A + A$	$\rightarrow a + A + A$
$\rightarrow a + a + A$	$\rightarrow a + a + A$
$\rightarrow a + a + a$	$\rightarrow a + a + a$

As another example, the grammar is ambiguous since there are two parse trees for the string a + a − a:



The language that it generates, however, is not inherently ambiguous; the following is a non-ambiguous grammar generating the same language:

$A \rightarrow A + a \mid A - a \mid a$

Minimization of Context Free Grammars:

In formal language theory, a context-free grammar (CFG) is a formal grammar in which every production rule is of the form

$A \rightarrow \alpha$

Where, A is a *single* nonterminal symbol, and α is a string of terminals and/or non-terminals (α can be empty). A formal grammar is considered "context free" when its production rules can be applied regardless of the context of a nonterminal. No matter which symbols surround it, the single nonterminal on the left hand side can always be replaced by the right hand side. This is what distinguishes it from a context-sensitive grammar.

Such a grammar has long lists of words, and also rules on what types of words can be added in what order. Higher rules combine several lower rules to make a sentence. Such sentences will be grammatically correct, but may not have any meaning. Each rule has its own symbol, which can be replaced with symbols representing lower rules, which can be replaced with words.

Chomsky normal form:

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 a 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)
Step 3	Remove unit productions. (Using the Unit production removal algorithm)
Step 4	Replace each production $A \rightarrow B_1 \ldots B_n$ where n > 2 with $A \rightarrow B_1C$ where $C \rightarrow B_2 \ldots B_n$. 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 $A \rightarrow aB$.

Problem:

Convert the following CFG into CNF

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

Solution:

(1) Since S appears in R.H.S, we add a new state S_0 and $S_0 \rightarrow S$ is added to the production set and it becomes −

$S_0 \rightarrow S, S \rightarrow ASA \mid aB, A \rightarrow B \mid S, B \rightarrow b \mid \varepsilon$

(2) Now we will remove the null productions −

$B \rightarrow \varepsilon$ and $A \rightarrow \varepsilon$

After removing $B \rightarrow \varepsilon$, the production set becomes −

$S_0 \rightarrow S, S \rightarrow ASA \mid aB \mid a, A \rightarrow B \mid S \mid \varepsilon, B \rightarrow b$

After removing $A \rightarrow \varepsilon$, the production set becomes −

$S_0 \rightarrow S, S \rightarrow ASA \mid aB \mid a \mid AS \mid SA \mid S, A \rightarrow B \mid S, B \rightarrow b$

(3) Now we will remove the unit productions.

After removing $S \rightarrow S$, the production set becomes −

$S_0 \rightarrow S, S \rightarrow ASA \mid aB \mid a \mid AS \mid SA, A \rightarrow B \mid S, B \rightarrow b$

After removing $S_0 \rightarrow S$, the production set becomes −

$S_0 \rightarrow ASA \mid aB \mid a \mid AS \mid SA, S \rightarrow ASA \mid aB \mid a \mid AS \mid SA$

$A \rightarrow B \mid S, B \rightarrow b$

After removing $A \rightarrow B$, the production set becomes −

$S_0 \rightarrow ASA \mid aB \mid a \mid AS \mid SA, S \rightarrow ASA \mid aB \mid a \mid AS \mid SA$

$A \rightarrow S \mid b$

$B \rightarrow b$

After removing $A \rightarrow S$, the production set becomes −

$S_0 \rightarrow ASA \mid aB \mid a \mid AS \mid SA, S \rightarrow ASA \mid aB \mid a \mid AS \mid SA$

$A \rightarrow b \mid ASA \mid aB \mid a \mid AS \mid SA, B \rightarrow b$

(4) Now we will find out more than two variables in the R.H.S

Here, $S_0 \rightarrow ASA, S \rightarrow ASA, A \rightarrow ASA$ violates two Non-terminals in R.H.S.

Hence, we will apply step 4 and step 5 to get the following final production set which is in CNF −

$S_0 \rightarrow AX \mid aB \mid a \mid AS \mid SA$

$S \rightarrow AX \mid aB \mid a \mid AS \mid SA$

$A \rightarrow b \mid AX \mid aB \mid a \mid AS \mid SA$

$B \rightarrow b$

$X \rightarrow SA$

(5) We have to change the productions $S_0 \rightarrow aB$, $S \rightarrow aB$, $A \rightarrow aB$

And the final production set becomes −

$S_0 \rightarrow AX \mid YB \mid a \mid AS \mid SA$

$S \rightarrow AX \mid YB \mid a \mid AS \mid SA$

$A \rightarrow b \mid AX \mid YB \mid a \mid AS \mid SA$

$B \rightarrow b$

$X \rightarrow SA$

$Y \rightarrow a$

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 $S' \rightarrow S$.

Step 2 Remove Null productions. (Using the Null production removal algorithm)

Step 3 Remove unit productions. (Using the Unit production removal algorithm)

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

Convert the following CFG into CNF

$S \rightarrow XY \mid X_n \mid p$

$X \rightarrow mX \mid m$

$Y \rightarrow Xn \mid o$

Solution:

Here, S does not appear on the right side of any production and there are no unit or null productions in the production rule set. So, we can skip Step 1 to Step 3.

Step 4:

Now after replacing

X in $S \rightarrow XY \mid Xo \mid p$

with

$mX \mid m$

We obtain

$S \rightarrow mXY \mid mY \mid mXo \mid mo \mid p.$

And after replacing

X in $Y \rightarrow X_n \mid o$

with the right side of

$X \rightarrow mX \mid m$

We obtain

$Y \rightarrow mXn \mid mn \mid o.$

Two new productions $O \rightarrow o$ and $P \rightarrow p$ are added to the production set and then we came to the final GNF as the following −

$S \rightarrow mXY \mid mY \mid mXC \mid mC \mid p$

$X \rightarrow mX \mid m$

$Y \rightarrow mXD \mid mD \mid o$

$O \rightarrow o$

$P \rightarrow p$

Pumping Lemma for Context Free Languages:

Lemma

If L is a context-free language, there is a pumping length p such that any string $w \in L$ of length $\geq p$ can be written as $w = uvxyz$, where $vy \neq \varepsilon$, $|vxy| \leq p$, and for all $i \geq 0$, $uvixyiz \in L$.

Applications of Pumping Lemma

Pumping lemma is used to check whether a grammar is context free or not. Let us take an example and show how it is checked.

Problem:

Find out whether the language $L= \{x^n y^n z^n \mid n \geq 1\}$ is context free or not.

Solution:

Let L is context free. Then, L must satisfy pumping lemma.

At first, choose a number n of the pumping lemma. Then, take z as $0^n 1^n 2^n$.

Break z into uvwxy, where

$|vwx| \leq n$ and $vx \neq \varepsilon$.

Hence vwx cannot involve both 0s and 2s, since the last 0 and the first 2 are at least (n+1) positions apart. There are two cases −

Case 1 − vwx has no 2s. Then vx has only 0s and 1s. Then uwy, which would have to be in L, has n 2s, but fewer than n 0s or 1s.

Case 2 − vwx has no 0s.

Here contradiction occurs.

Hence, L is not a context-free language.

Enumeration of properties of CFL (proofs omitted):

Context-free languages are closed under −

- Union
- Concatenation
- Kleene Star operation

Context-free languages are not closed under −

- Intersection
- Intersection with Regular Language
- Complement