ABHYASMITRA.IN
STUDY NOTES

THEORY OF COMPUTATIONS

Unit III — Context Free Grammars (CFG) and Languages

Topics Covered (09 Hours)

Formal Definition of Context Free Grammar (CFG)
Leftmost & Rightmost Derivations & Sentential Forms
Derivation Trees / Parse Trees & Context Free Languages (CFL)
Ambiguous Grammars & Resolving Grammar Ambiguity
CFG Simplification: Eliminating Epsilon & Unit Productions
CFG Simplification: Eliminating Useless Productions & Symbols
Normal Forms: Chomsky Normal Form (CNF) & GNF Conversion
Pumping Lemma for Context Free Languages (Non-CFL Proofs)
Closure Properties of CFLs & The Chomsky Hierarchy
Applications of CFGs: Palindromes, Parsers, XML, and DTDs

Compiled in a simple, easy-to-understand, book style format
for students of Computer Science and Engineering

TABLE OF CONTENTS

3.1 Context Free Grammars & Derivations

While regular languages are useful for describing simple tokens (like identifiers or numbers in a programming language), they cannot represent nested structures (like matching parentheses or block structures). To define these more complex, nested languages, we use Context Free Grammars (CFGs).

Context Free Grammar Formal Definition

A Context Free Grammar is formally defined as a 4-tuple:

$G = (V, \Sigma, R, S)$

Where:

1. V is a finite set of Variables (also called Non-Terminals).

2. Σ is a finite set of Terminal symbols, disjoint from V ($\Sigma \cap V = \emptyset$).

3. R is a finite set of Production Rules, mapping a Variable to a string of variables and terminals (mapping $V \rightarrow (V \cup \Sigma)^*$).

4. $S \in V$ is the start Variable.

Derivations & Parse Trees

A derivation is the sequence of production steps used to generate a string from the start variable S. If we replace variables from left to right, it is a Leftmost Derivation. If we replace variables from right to left, it is a Rightmost Derivation. A Sentential Form is any string of terminals and/or variables generated during the derivation.

A Derivation Tree (or Parse Tree) is a graphical representation of a derivation. The root node is the start symbol S, internal nodes are variables, and leaf nodes are terminals. The yield of the tree is the string formed by reading the leaves from left to right.

3.2 Grammar Simplification & Normal Forms

To construct compilers and parsers efficiently, context-free grammars must be cleaned of redundant rules and converted into standard mathematical structures called Normal Forms.

CFG Simplification Steps

- **1. Eliminating Epsilon (ε) Productions:** An ε-production has the form A → ε. We find all variables (Nullable Variables) that can eventually derive ε. For every production rule containing a nullable variable, we add duplicate rules with that variable omitted, and then delete the raw ε-productions.
- **2. Eliminating Unit Productions:** A unit production has the form A → B, where both are variables. These add redundant steps. We build a dependency graph of unit rules and substitute the terminal derivations of B directly into A, then delete the unit rules.
- **3. Eliminating Useless Productions & Symbols:** Useless symbols are variables or terminals that either cannot derive a string of terminals (Non-generating symbols) or cannot be reached from the start symbol S (Non-reachable symbols). We find and prune them.

Chomsky Normal Form (CNF)

A grammar is in Chomsky Normal Form if all production rules are of the form:

$$A \rightarrow BC \quad or \quad A \rightarrow a$$

Where A, B, and C are variables, and 'a' is a terminal symbol. Any CFG (without ε) can be converted to CNF. The conversion algorithm replaces long production chains (like A → BCD) with binary variable trees (e.g., A → BX and X → CD) and introduces variables to isolate terminals.

Greibach Normal Form (GNF)

A grammar is in Greibach Normal Form if all production rules are of the form:

$$A \rightarrow a\alpha$$

Where 'a' is a single terminal symbol, and α is a string of zero or more variables. GNF is highly useful for parsing because every production step immediately generates exactly one terminal symbol.

3.3 CFL Properties & The Chomsky Hierarchy

Like regular languages, context-free languages have limits. We use the Pumping Lemma for CFLs to prove a language is not context-free.

Pumping Lemma for CFLs

The Pumping Lemma for CFLs states that if a language L is context-free, there exists a pumping length p such that any string z ∈ L with length |z| ≥ p can be split into five substrings, z = uvwxy, satisfying:

- 1. |vwx| ≤ p (the middle section must be within the pumping length).
- 2. vx != ε (i.e., at least one of the pumped strings v or x must be non-empty).
- 3. For all i ≥ 0, u(v^i)w(x^i)y ∈ L (v and x can be pumped simultaneously, keeping the balance).

Using this lemma, we can prove that L = {a^n b^n c^n | n ≥ 0} is NOT context-free, because any split uvwxy cannot pump 'a's, 'b's, and 'c's in equal proportions.

The Chomsky Hierarchy

Noam Chomsky classified all formal grammars into a hierarchy of four nested classes:

- **Type 0 (Unrestricted Grammars):** No restrictions on rules (α → β). Accepted by Turing Machines.
- **Type 1 (Context-Sensitive Grammars):** Rules must be length-increasing (|α| ≤ |β|). Accepted by Linear Bounded Automata.
- **Type 2 (Context-Free Grammars):** Left side must be a single variable (A → α). Accepted by Pushdown Automata.
- **Type 3 (Regular Grammars):** Left side is a single variable, right side is a terminal and/or variable (A → aB or A → a). Accepted by Finite Automata.

Unit Summary

This unit discussed Context Free Grammars and the Chomsky Hierarchy.

- Context Free Grammars define Context Free Languages using variables and production rules.
- Parse trees represent leftmost and rightmost derivations graphically.
- Simplification algorithms eliminate epsilon rules, unit rules, and useless symbols.
- CFGs can be converted to Chomsky Normal Form (CNF) and Greibach Normal Form (GNF).
- The Pumping Lemma for CFLs is used to prove a language is not context-free.
- The Chomsky Hierarchy organizes formal grammars into four distinct mathematical levels.