# TUTORIAL SESSION 19:

## Chomsky Hierarchy and Halting Problem

### Concept Building

### Chomsky Hierarchy

The Chomsky hierarchy is a classification of formal grammars that describes the relationship between different types of languages and the computational power required to recognize them. It consists of four levels, each corresponding to a different class of languages and types of automata:

1. **Type 0: Unrestricted Grammars**
   - **Definition**: These grammars have no restrictions on their production rules. They can generate recursively enumerable languages.
   - **Equivalent Automaton**: Turing Machines (TMs).
   - **Example**: Any language that can be recognized by a Turing machine, such as the set of all strings over the alphabet {0, 1}.
2. **Type 1: Context-Sensitive Grammars**
   - **Definition**: These grammars have production rules of the form $\alpha A\beta \rightarrow \alpha w\beta$, where A is a non-terminal, and w is a string of terminals and non-terminals. The length of w must be greater than or equal to that of $\alpha A\beta$.
   - **Equivalent Automaton**: Linear Bounded Automata (LBA).
   - **Example**: The language $\{a^n b^n c^n \mid n \geq 1\}$.
3. **Type 2: Context-Free Grammars**
   - **Definition**: These grammars have production rules of the form $A \rightarrow \alpha A \rightarrow \alpha$, where AA is a single non-terminal and $\alpha\alpha$ is a string of terminals and non-terminals.
   - **Equivalent Automaton**: Pushdown Automata (PDA).
   - **Example**: The language $\{a^n b^n \mid n \geq 0\}$.
4. **Type 3: Regular Grammars**
   - **Definition**: These grammars have production rules of the form $A \rightarrow aB$ or $A \rightarrow a$, where A and B are non-terminals and aa is a terminal.
   - **Equivalent Automaton**: Finite Automata (FA).
   - **Example**: The language $\{a^* \mid n \geq 0\}$.

The hierarchy is structured such that each level is a superset of the levels below it. For example, all regular languages (Type 3) are also context-free (Type 2), context-sensitive (Type 1), and recursively enumerable (Type 0). This hierarchy helps in understanding the computational complexity and capabilities of different types of languages and the machines that recognize them.

**Halting Problem**

The Halting Problem is a fundamental concept in computability theory that addresses the question of whether a given Turing machine will halt (i.e., finish its computation) on a given input or will run indefinitely.

- **Definition**: The Halting Problem can be formally stated as follows: Given a Turing machine MM and an input string ww, determine whether MM halts when run with input ww.
- **Undecidability**: Alan Turing proved that there is no general algorithm that can solve the Halting Problem for all possible Turing machines and inputs. This means that it is impossible to construct a Turing machine (or any computational model) that can correctly determine whether any arbitrary Turing machine will halt on a given input.
- **Implications**: The undecidability of the Halting Problem has profound implications for computer science:
    - It establishes limits on what can be computed algorithmically.
    - It shows that certain problems cannot be solved by any algorithm, regardless of how powerful the machine is.
    - It leads to a deeper understanding of the boundaries between decidable and undecidable problems in computational theory.

## Pre-Tutorial (To be completed by student before attending tutorial session)

**1. Give an example of a language that is context-free but not regular.**

**Solution:**

**L = {a^n b^n | n ≥ 0}**

This is context-free but not regular, as it fails the pumping lemma.

**2. Is the language $L = \{a^i b^j c^k \mid i \neq j, j \neq k\}$ regular? If no, find the type of this langauge.**

**Solution:**

The language **L = {a^i b^j c^k | i ≠ j, j ≠ k}** is **not regular** but **context-free**. Regular languages can't handle such dependencies.

## IN-TUTORIAL (To be carried out in presence of faculty in classroom)

**1. Consider a Turing machine that can determine whether another Turing machine halts on a given input. If such a machine exists, what can be the implications?**

**Solution:**

If such a machine existed, it would contradict **Turing's Halting Theorem**, proving that the Halting Problem is undecidable.

**2. Give a problem that is equivalent to halting problem.**

**Solution:**

The **Post Correspondence Problem (PCP)** is equivalent to the Halting Problem and is undecidable.

**3. Describes the relationship between the Halting Problem and recursive functions.**

**Solution:**

- The Halting Problem shows no recursive function can decide halting for all inputs.
- It proves that not all recursive functions are decidable.
- Some recursive functions are decidable, but halting is undecidable in general.

**4. If we can create a Turing machine that solves the Halting Problem for a specific class of programs, what can we conclude?**

**Solution:**

- The class of programs is **decidable**.
- The Halting Problem is solvable for that class.
- The class has **restricted behavior** allowing decidability.

## Post-Tutorial (To be carried out by student after attending tutorial session)

**1. Compare and contrast Type 1 and Type 2 languages in terms of their generative grammars and recognizers.**

**Solution:**

- **Type 1 (Context-Sensitive):** Generated by context-sensitive grammars, recognized by linear-bounded automata.
- **Type 2 (Context-Free):** Generated by context-free grammars, recognized by pushdown automata.
- **Comparison:** Type 1 is more powerful, with more complex recognition requirements than Type 2.

**2. Illustrate the relationship between the Chomsky Hierarchy and computational complexity.**

**Solution:**

1. **Type 3 (Regular):** Recognized by finite automata, efficient (linear time).
2. **Type 2 (Context-Free):** Recognized by pushdown automata, polynomial time.
3. **Type 1 (Context-Sensitive):** Recognized by linear-bounded automata, more complex.
4. **Type 0 (Recursively Enumerable):** Recognized by Turing machines, undecidable.

**3. How does the concept of language containment within the Chomsky Hierarchy help in understanding the capabilities of different computational models?**

**Solution:**

- **Containment:** Higher levels (e.g., Type 0) contain lower levels (e.g., Type 3).
- **Computational Power:** More complex models (Turing machines) recognize more languages.
- **Limitations:** Simpler models (finite automata, PDAs) handle fewer, simpler languages.
- **Understanding Models:** Shows the relationship between language complexity and model capabilities.

| | | | |
|---|---|---|---|
| Experiment # | <TO BE FILLED BY STUDENT> | Student ID | <TO BE FILLED BY STUDENT> |
| Date | <TO BE FILLED BY STUDENT> | Student Name | <TO BE FILLED BY STUDENT> |

**Viva - Questions**

1. Explain the characteristics of context-free grammars (CFGs) and the types of languages they generate.

**Solution:**

Context-free grammars generate **context-free languages**, with production rules of the form **A → γ**.

2. How do regular languages relate to context-free languages in the Chomsky hierarchy?

**Solution:**

Regular languages are a subset of context-free languages, with simpler rules and less computational power.

(For Evaluator's use only)

| Comment of the Evaluator (if Any) | Evaluator's Observation |
|---|---|
| | Marks Secured: out of **50**<br>Full Name of the Evaluator:<br>Signature of the Evaluator Date of<br>Evaluation: |