CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL IPN
UNIDAD TAMAULIPAS

# Assignment #3

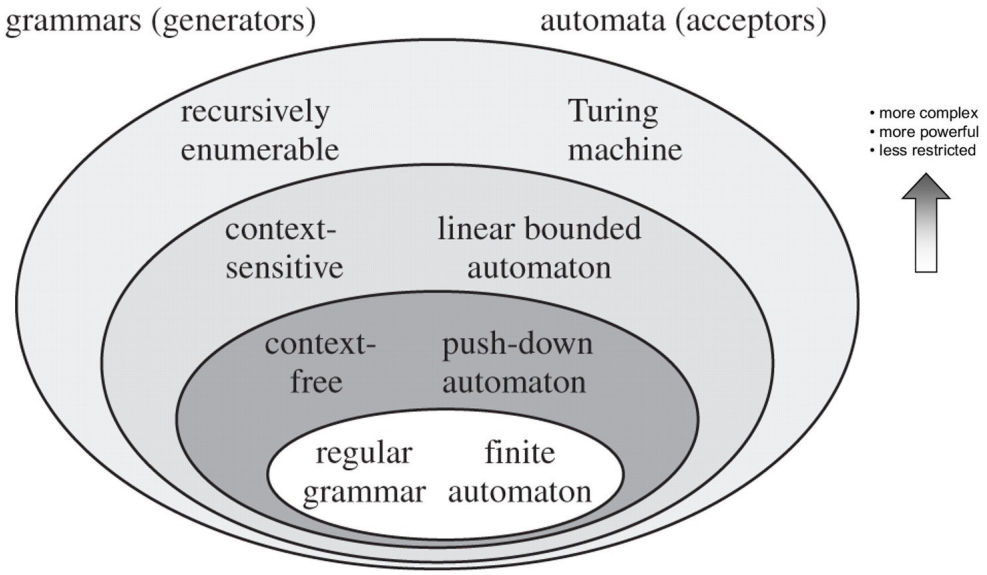
Student: *Luis Alberto Ballado Aradias*
Course: *Tecnologías Computacionales (Sep - Dec 2022)*
Professor: *Dr. Edwin Aldana Bobadilla*
*October 13, 2022*

## Chomsky Hierarchy

Any language is a structured medium of communication no matter if it is spoken or written natural language, coded or sign language, or a formal programming language such as Python or Java. Languages are characterised by two basic elements: **syntax** (grammatical rules) and **semantics** (meaning). In language like the way we communicate each other, the meaning might vary depending upon a third factor **context of usage**.

Depending on restrictions and complexity present in the grammar, languages find a place in the hierarchy of formal languages. **Noam Chomsky**, an American linguist cum cognitive scientist, defined this hierarchy in 1956 and hence it is calles **Chomsky Hierarchy**.

Although his concept is quite old, there is renewed interest because of it is relevance to Natural Language Processing. Chomsky Hierarchy help us answer question like **Can a natural language like Spanish be described .. parsed, or even compiled** with the same methods as used for formal or artificial languages in Computer Science?



According to Chomsky hierarchy, grammar is divided into 4 types:

1. Regular Grammar
2. Context Free Grammar (deterministic and non deterministic)
3. Context Sesitive Grammar
4. Recursively-enumerable Grammar

### Regular Grammar

Most restrictive of the set, they generate regular languages. They must have a single non-terminal on the left-hand-side and a right-hand-side consisting of a single terminal or single terminal followed by a single non-terminal.

**Example:**

Regex to define tokens such as identifiers, language keywords in programming languages. A coin vending machine that accepts only 1-Rupee, 2-Rupee and 5-Rupee coins has a regular language with only three words – 1, 2, 5.

### Context-Free Grammar

Generate context-free languages, a category of immense interest to NLP practitioners. Here all rules take the form $A \implies \beta$, where A is a single non-terminal symbol and $\beta$ is a string of symbols.

**Example:**

Statement blocks in programming languages such as functions in parentheses, If-Else, for loops. In natural language, nouns and their plurals can be recognized through one Non Deterministic Finite Automaton (NFA), verbs and their different forms can be recognized through another NFA, and then combined.
Singular (The girl runs home $\implies$ Girl + Runs).
Plural (The girls run home $\implies$ Girls + Run)

### Context-Sensitive Grammar

The highest programmable level, they generate context-sensitive languages. They have rules of the form $\alpha A \beta \implies \alpha \gamma \beta$ with A as a non-terminal and $\alpha$, $\beta$, $\gamma$ as strings of terminals and non-terminals. Strings $\alpha$, $\beta$ may be empty, but $\gamma$ must be nonempty.

**Example:**

Though most language constructs in natural language are context-free, in some situations linear matching of tokens has to be done, such as - The square roots of 16, 9

and 4 are 4, 3 and 2, respectively. Here 16 is to be matched with 4, 9 is matched with 3, and 4 is matched with 2.

### Recursively enumerable grammar

Are too generic and unrestricted to describe the syntax of either programming or natural languages

**Example:**

A language with no restrictions is not conducive to communication or automation. Hence there are no common examples for this type. However, some mathematical seemingly unsolvable equations are expressed in this form.

## Regular Expression in Real Application

Exercise: Create a regular expression to accept vehicle registration plates of Mexico State. There are three-letter series assigned **LGA-PEZ**


```
/^[L-P]{1}[G-ZA-E]{1}[A-Z]{1}-\d{2}-\d{2}$/
```

Explanation:

- ∧ asserts position at start of the string
- Match a single character present in the list below [***L*** – ***P***]{**1**} matches the previous token exactly **one** time (meaningless quantifier)
  [***L*** – ***P***] matches a single character in the range between L and P (case sensitive)
- Match a single character present in the list below [***G*** – ***ZA*** – ***E***]{**1**} matches the previous token exactly **one** time (meaningless quantifier)
  [***G*** – ***Z***] matches a single character in the range between G and Z (case sensitive)
- Match a single character present in the list below [***A*** – ***Z***]{**1**} matches the previous token exactly one time (meaningless quantifier)
  [***A*** – ***Z***] matches a single character in the range between A and Z (case sensitive)
- ***d*** matches a digit (equivalent to [0-9])
  {**2**} matches the previous token exactly 2 times
- ***d*** matches a digit (equivalent to [0-9])
  {**2**} matches the previous token exactly 2 times
- **$** asserts position at the end of the string, or before the line terminator right at the end of the string (if any)