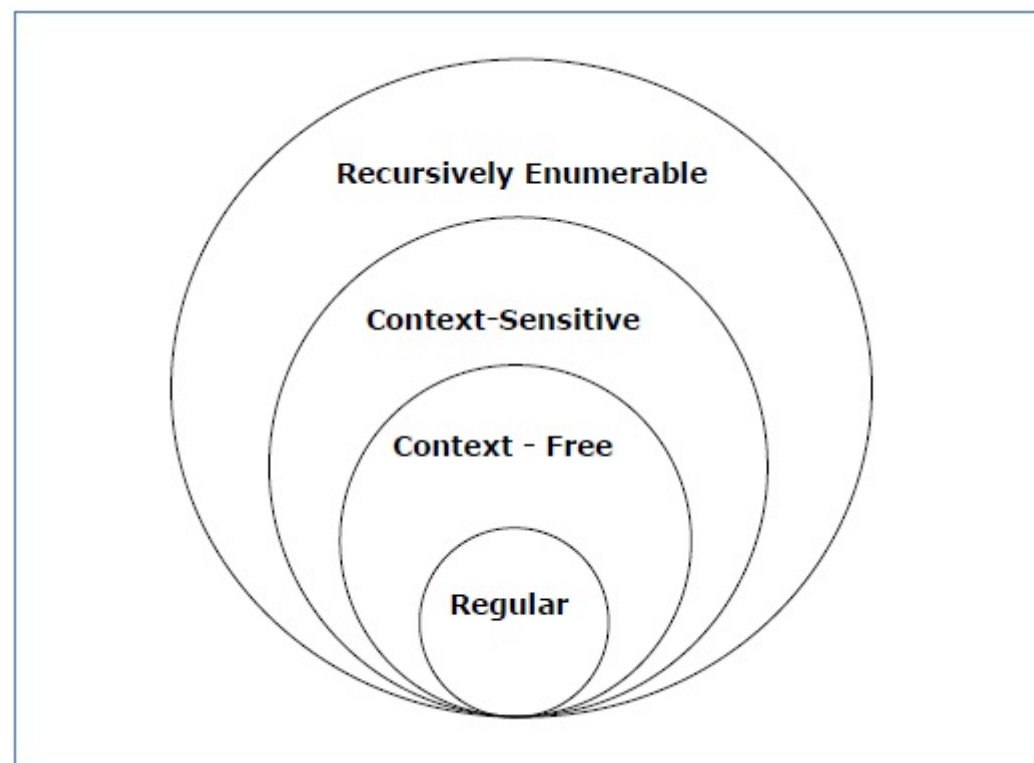


# Chomsky Classification of Grammars

According to Noam Chomsky, there are four types of grammars – Type 0, Type 1, Type 2, and Type 3. The following table shows how they differ from each other –

Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type 0	Unrestricted grammar	Recursively enumerable language	Turing Machine
Type 1	Context-sensitive grammar	Context-sensitive language	Linear-bounded automaton
Type 2	Context-free grammar	Context-free language	Pushdown automaton
Type 3	Regular grammar	Regular language	Finite state automaton

Take a look at the following illustration. It shows the scope of each type of grammar –



## Type - 3 Grammar

**Type-3 grammars** generate regular languages. Type-3 grammars 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.

The productions must be in the form  $\mathbf{X} \rightarrow \mathbf{a}$  or  $\mathbf{X} \rightarrow \mathbf{aY}$

where  $\mathbf{X}, \mathbf{Y} \in \mathbf{N}$  (Non terminal)

and  $\mathbf{a} \in \mathbf{T}$  (Terminal)

The rule  $\mathbf{S} \rightarrow \epsilon$  is allowed if  $\mathbf{S}$  does not appear on the right side of any rule.

### Example

```
X → ε
X → a | aY
Y → b
```

## Type - 2 Grammar

**Type-2 grammars** generate context-free languages.

The productions must be in the form  $\mathbf{A} \rightarrow \mathbf{\gamma}$

where  $\mathbf{A} \in \mathbf{N}$  (Non terminal)

and  $\mathbf{\gamma} \in (\mathbf{T} \cup \mathbf{N})^*$  (String of terminals and non-terminals).

These languages generated by these grammars are recognized by a non-deterministic pushdown automaton.

### Example

```
S → X a
X → a
X → aX
X → abc
X → ε
```

## Type - 1 Grammar

**Type-1 grammars** generate context-sensitive languages. The productions must be in the form

$$\alpha \mathbf{A} \beta \rightarrow \alpha \mathbf{\gamma} \beta$$

where  $\mathbf{A} \in \mathbf{N}$  (Non-terminal)

and  $\alpha, \beta, \mathbf{\gamma} \in (\mathbf{T} \cup \mathbf{N})^*$  (Strings of terminals and non-terminals)

The strings  $\alpha$  and  $\beta$  may be empty, but  $\mathbf{\gamma}$  must be non-empty.

The rule  $\mathbf{S} \rightarrow \epsilon$  is allowed if  $\mathbf{S}$  does not appear on the right side of any rule. The languages generated by these grammars are recognized by a linear bounded automaton.

### Example

$$AB \rightarrow AbBc$$
$$A \rightarrow bcA$$
$$B \rightarrow b$$

## Type - 0 Grammar

**Type-0 grammars** generate recursively enumerable languages. The productions have no restrictions. They are any phase structure grammar including all formal grammars.

They generate the languages that are recognized by a Turing machine.

The productions can be in the form of  $\alpha \rightarrow \beta$  where  $\alpha$  is a string of terminals and nonterminals with at least one non-terminal and  $\alpha$  cannot be null.  $\beta$  is a string of terminals and non-terminals.

### Example

$$S \rightarrow ACaB$$
$$Bc \rightarrow acB$$
$$CB \rightarrow DB$$
$$aD \rightarrow Db$$