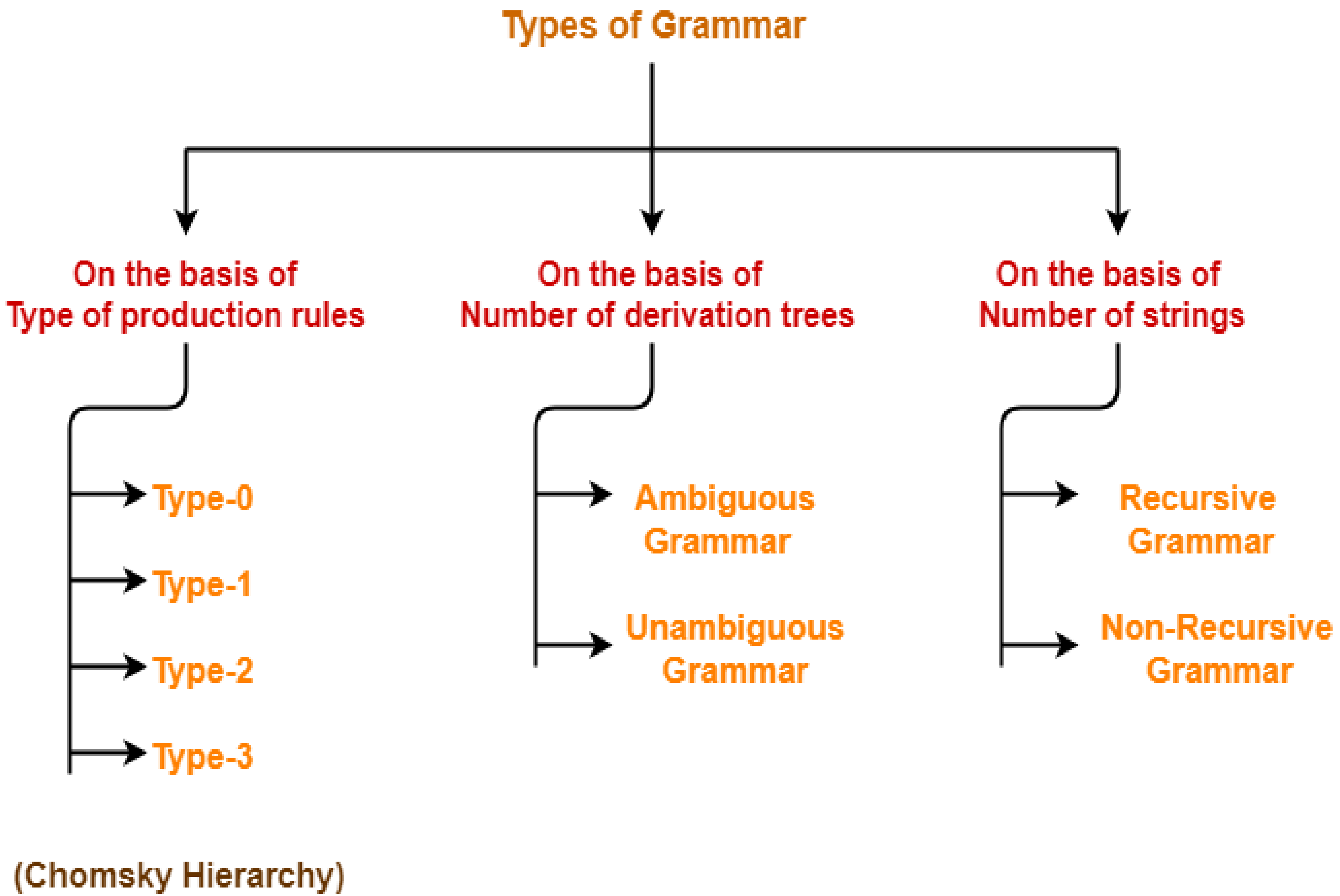
Chomsky Hierarchy and Closure Properties of CFL

---Sakshi Surve

About Chomsky :

- Linguistics have attempted to define grammars since the inception of natural languages like English, Sanskrit, Mandarin, etc.
- Noam Chomsky was a Philosopher of Languages
- He was a Professor of Linguistics
- **Noam Chomsky** gave a mathematical model of grammar in 1956 which is effective for writing computer languages.
- The theory of formal languages finds its applicability extensively in the fields of Computer Science……Formally called as Chomsky Hierarchy


Types of Grammar
On the basis of
Type of production rules
On the basis of
Number of derivation trees
On the basis of
Number of strings
Type-0
Type-1
Type-2
Type-3
Ambiguous
Grammar
Unambiguous
Grammar
Recursive
Grammar
Non-Recursive
Grammar
(Chomsky Hierarchy)

- (LHS) a → b (RHS) is a production rule.
- And, that is the basis of classification in Chomsky Hierarchy

Chomsky Hierarchy :

- Comprises four types of languages and their associated grammars and machines.
  - Type 3: Regular Languages
  - Type 2: Context-Free Languages
  - Type 1: Context-Sensitive Languages
  - Type 0: Recursively Enumerable Languages

Type 3 : Regular 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 **X → a or X → aY**

  where **X, Y ∈ N** (Non terminal)

  and **a ∈ T** (Terminal)

- These languages generated by these grammars are be recognized by a **Finite Automaton.**

- **Example**

$$X \rightarrow \varepsilon, \; X \rightarrow a \mid aY, \; Y \rightarrow b$$

Type 2 : Context Free Grammar

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

- The productions must be in the form **A → γ**

  where **A ∈ N** (Non terminal)

  and **γ ∈ (T ∪ N)*** (String of terminals and non-terminals).

- These languages generated by these grammars are be recognized by a **Pushdown Automaton.**

- **Example :**

  S → X a X → a X → aX X → abc X → ε

Type 1 : Context Sensitive Grammar

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

$$\alpha\, A\, \beta \rightarrow \alpha\, \gamma\, \beta$$

where $A \in N$ (Non-terminal)

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

- The strings α and β may be empty, but γ must be non-empty.
- |LHS| <=|RHS| productions
- The start variable S cannot appear on the RHS
- The languages generated by these grammars are recognized by a **Linear bounded automaton.**
- **Example**

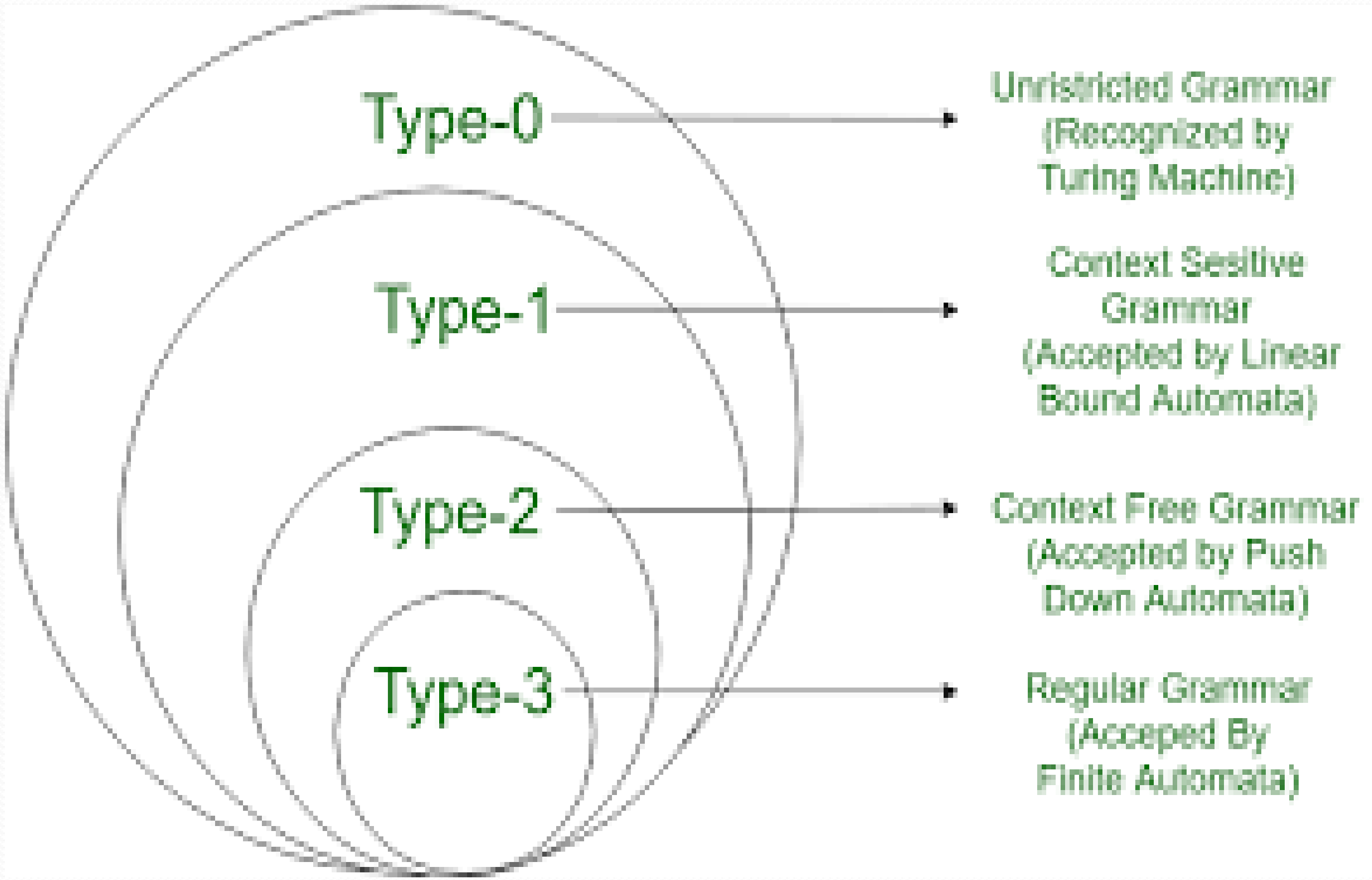
$AB \rightarrow AbBc$, $A \rightarrow bcA$, $B \rightarrow b$

Type 0 : Unrestricted Grammar

- **Type-o 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 $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ where $\boldsymbol{\alpha}$ is a string of terminals and non terminals with at least one non-terminal and $\boldsymbol{\alpha}$ cannot be null. $\boldsymbol{\beta}$ is a string of terminals and non-terminals.
- **Example**

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

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


Type-0
Type-1
Type-2
Type-3
Unristricted Grammar (Recognized by Turing Machine)
Context Sesitive Grammar (Accepted by Linear Bound Automata)
Context Free Grammar (Accepted by Push Down Automata)
Regular Grammar (Acceped By Finite Automata)

Closure Properties of CFL

Closure properties of CFL

- **Closure properties** consider operations on CFL that are guaranteed to produce a CFL

- The CFL's are closed under *union*, *concatenation*, *closure*

Union

- Let L_1 and L_2 be two context free languages. Then $L_1 \cup L_2$ is also context free.
- Use $L = \{a, b\}$, $s(a) = L_1$ and $s(b)=L_2$.$s(L)= L_1 \cup L_2$
- To get grammar for $L_1 \cup L_2$?
  - Add new start symbol S and rules $S \rightarrow S_1|S_2$
  - We get grammar G = (V, T, P, S) where
    $V = V_1 \cup V_2 \cup \{ S \}$, where $S \notin V_1 \cup V_2$
    $P = P_1 \cup P_2 \cup \{ S \rightarrow S_1 \mid S_2 \}$
- Example:
  - $L_1 = \{ a^n b^n \mid n \geq 0 \}$, $L_2 = \{ b^n a^n \mid n \geq 0 \}$
  - $G_1 : S_1 \rightarrow aS_1b \mid \varepsilon$, $G_2 : S_2 \rightarrow bS_2a \mid \varepsilon$
  - $L1 \cup L2$ is $G = (\{S_1, S_2, S\}, \{a, b\}, P, S)$ where $P = \{P1 \cup P2 \cup \{S \rightarrow S_1 \mid S_2 \}\}$

- Example
- Let $L_1 = \{ a^n b^n , n > 0\}$. Corresponding grammar G_1 will have P: $S1 \rightarrow aAb|ab$
- Let $L_2 = \{ c^m d^m , m \geq 0\}$. Corresponding grammar G_2 will have P: $S2 \rightarrow cBb|\ \varepsilon$
- Union of L_1 and L_2, $L = L_1 \cup L_2 = \{ a^n b^n \} \cup \{ c^m d^m \}$
- The corresponding grammar G will have the additional production $S \rightarrow S1 \mid S2$

Concatenation

- If L_1 and L_2 are context free languages, then L_1L_2 is also context free.
- Let L={ab}, s(a)=L_1 and s(b)=L_2. Then s(L)=L_1L_2
- To get grammar for L_1L_2 ?
  - Add new start symbol and rule $S \to S_1S_2$
  - We get G = (V, T, P, S) where

    $V = V_1 \cup V_2 \cup \{ S \}$, where $S \notin V_1 \cup V_2$

    $P = P_1 \cup P_2 \cup \{ S \to S_1S_2 \}$
- Example:
  - $L_1 = \{ a^nb^n \mid n \geq 0 \}$ with G_1: $S_1 \to aS_1b \mid \varepsilon$
  - $L_2 = \{ b^na^n \mid n \geq 0 \}$ with G_2 : $S_2 \to bS_2a \mid \varepsilon$
  - $L_1L_2 = \{ a^nb^{\{n+m\}}a^m \mid n, m \geq 0 \}$ with $G = (\{S, S_1, S_2\}, \{a, b\}, \{S \to S_1S_2, S_1 \to aS_1b \mid \varepsilon, S_2 \to bS_2a\}, S)$

- Example
- Concatenation of the languages L_1 and L_2,

  $L = L_1L_2 = \{ a^n b^n c^m d^m \}$
- The corresponding grammar G will have the additional production $S \rightarrow S1\ S2$

Kleene Closure

- If L is a context free language, then L^* is also context free.
- Use $L=\{a\}^*$ or $L=\{a\}^+$, $s(a)=L_1$. Then $s(L)=L_1^*$
- Example:
  - Let $L = \{ a^n b^n , n \geq 0\}$. Corresponding grammar G will have P: $S \rightarrow aAb \mid \varepsilon$
  - Kleene Star $L_1 = \{ a^n b^n \}^*$
  - The corresponding grammar G_1 will have additional productions $S1 \rightarrow SS_1 \mid \varepsilon$

- To get grammar for $(L_1)^*$
  - Add new start symbol S and rules $S \rightarrow SS_1 \mid \varepsilon$.
  - We get $G = (V, T, P, S)$ where

    $V = V_1 \cup \{ S \}$, where $S \notin V_1$

    $P = P_1 \cup \{ S \rightarrow SS_1 \mid \varepsilon\}$

Ambiguous Grammar	Unambiguous Grammar
A grammar is said to be ambiguous if for at least one string generated by it, it produces more than one- •parse tree •or derivation tree •or syntax tree •or leftmost derivation •or rightmost derivation	A grammar is said to be unambiguous if for all the strings generated by it, it produces exactly one- •parse tree •or derivation tree •or syntax tree •or leftmost derivation •or rightmost derivation
For ambiguous grammar, leftmost derivation and rightmost derivation represents different parse trees.	For unambiguous grammar, leftmost derivation and rightmost derivation represents the same parse tree.
Ambiguous grammar contains less number of non-terminals.	Unambiguous grammar contains more number of non-terminals.

For ambiguous grammar, length of parse tree is less.	For unambiguous grammar, length of parse tree is large.
Ambiguous grammar is faster than unambiguous grammar in the derivation of a tree. (Reason is above 2 points)	Unambiguous grammar is slower than ambiguous grammar in the derivation of a tree.
Example- $E \rightarrow E + E \;/\; E \times E \;/\; id$ (Ambiguous Grammar)	**Example-** $E \rightarrow E + T \;/\; T$ $T \rightarrow T \times F \;/\; F$ $F \rightarrow id$ (Unambiguous Grammar)