**The Chomsky Hierarchy**

**Grammar:**

A grammar ‘G’ can be formally described using 4 tuples as G= (V, T, S, P) where,

V= Set of variables or non-terminal (denoted by uppercase letters) symbols

T= Set of terminal (denoted by lowercase letters) symbols

S= Start symbol

P= Production rules for terminals and non-terminals

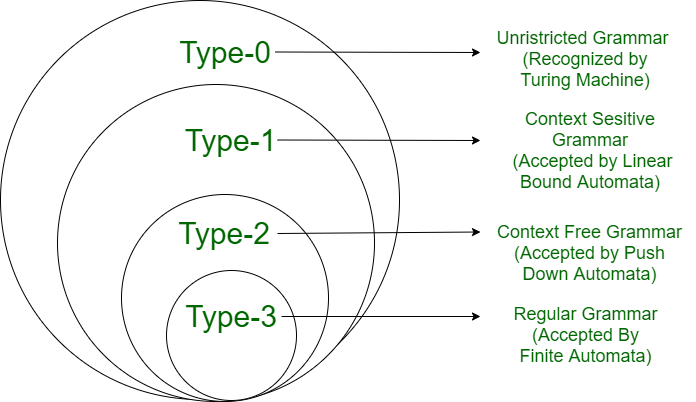
A production rule has the form α→ꞵ where α and ꞵ are strings on V U T (U= union operation) and at least one symbol of α belongs to V.

Example: G= ({S, A, B}, {a, b}, S, {S→AB, A→a, B→b})

**The Noam Chomsky Hierarchy:**

Chomsky Hierarchy represents the class of languages that are accepted by different machines. The category of language in Chomsky's Hierarchy is as given below:

1. Type 0 is known as Unrestricted Grammar.
2. Type 1 is known as Context Sensitive Grammar.
3. Type 2 is known as Context-Free Grammar.
4. Type 3 is known as Regular Grammar.



**Fig: Chomsky Hierarchy**

**Type-0 Grammar:**

Type 0 grammar is known as unrestricted grammar. There is no restriction on the grammar rules of these types of languages. It includes all formal grammar. These languages can be efficiently modeled or recognized by Turing machines. It is also known as the recursively enumerable languages.

These are defined by rules of the form α→ꞵ, where α ∈ (V U T) ⁺ and ꞵ ∈ (V U T) ̽ or α ∈ (V + T) ̽ V (V + T) ̽ and ꞵ ∈ (V + T) ̽ and must be at least one variable on the left side of production (α ≠ null).

**For example:**

The grammar with terminals {a, b, c, d}, non-terminals or values {S, A, B, C, D} then production rules for type-0:

1) S→aBc

2) aB→cA

3) Ac→d

**Type-1 Grammar:**

Type 1 grammar is known as Context Sensitive Grammar. Context-sensitive grammar is used to represent context-sensitive language. It is recognized by the linear bound automata. The context-sensitive grammar follows the following rules:

* The context-sensitive grammar may have more than one symbol on the left-hand side of their production rules.
* The number of symbols on the left-hand side must not exceed the number of symbols on the right-hand side.
* The rule of the form A → ε is not allowed unless A is a start symbol. It does not occur on the right-hand side of any rule.
* The Type 1 grammar should be Type 0. In type 1, Production is in the form of V → T where the count of symbols in V is less than or equal to T.

These are defined by rules of the form α→ꞵ, such that |α| ≤ | ꞵ| where α, ꞵ ∈ (V U T) ⁺.

**For example:**

The grammar with terminals {a, b, c}, non-terminals or values {S, A, B, C} then production rules for type-1:  
1) S→abc | aAbc

2) Ab→bA

3) Ac→Bbcc

4) bB→Bb

5) aB→aa | aaA

**Type-2 Grammar:**

Type 2 Grammar is known as Context-Free Grammar. Context-free languages are the languages that can be represented by context-free grammar (CFG). Type 2 should be type 1.

These are defined by rules of the form α→ꞵ, where α ∈ V left-hand side of production can have only one variable and ꞵ ∈ (V + T) ̽.

**For example:**

The grammar with terminals {a, b}, non-terminals or values {S, A, B} then production rules for type-2:

1) S→AB

2) A→a

3) B→b

**Type-3 Grammar:**

Type 3 Grammar is known as Regular Grammar. Regular languages are those languages that can be described using regular expressions. These languages can are exactly all languages that can be accepted by a finite-state automaton (NFA or DFA).

Type 3 is the most restricted form of grammar. The Type 3 grammar should be Type 2 and Type 1. Type 3 should be in the form of V → T\*V / T\* or, V → V T\* / T\* means α→VT (A non-terminal variable giving a combination of non-terminal and terminal e.g. S→ aB | Ba, S→aaB | Baa, S→abB | Bab) or α→T (A non-terminal variable giving only a terminal e.g. S→a | b).

Regular grammar can be divided into two types:

1. **Right regular grammar:** A grammar is said to be right linear if all productions are of the form A→xB, A→x where A, B ∈ V and x ∈ T.

**For example:** The grammar with terminals {a, b}, non-terminals or values {S, A, B} then production rules for type-3: S→abS | b.

1. **Left regular grammar:** A grammar is said to be left linear if all productions are of the form A→Bx, A→x where A, B ∈ V and x ∈ T.

**For example:** The grammar with terminals {a, b}, non-terminals or values {S, A, B} then production rules for type-3: S→Sab | b.

**Identify the type of grammar:**

The main restricted points for 4 types of grammar is given below:

α→ꞵ (left portion gives right portion) commons for all type of grammar.

\* Type- 0: α ≠ null

\* Type- 1: |α| ≤ | ꞵ|

\* Type- 2: α ∈ V

\* Type- 3: α→aB | a | b | Ba | Ab | bA | abS | Sab

Type- 0 is the less restricted and type- 3 is the most restricted grammar cause type- 3 contains all other types of restrictions. So, when all given products will be valid for type- 3 than those products are also valid for other types.

**Question-1:**

The grammar with terminals {a, b, c, d}, non-terminals or values {S, A, B, C, D}. Then given products follow which type(s) of grammar?

S→ACaB

Bc→acB

CB→DB

aD→Db

**Solution:**

Only when all products are valid for a type of grammar then we can say valid for that type of grammar.

First we should check from type-3 grammar but mentioned products violate the type-3 grammar restrictions cause for type-3 we know that is valid only when one variable gives only a terminal or when one variable gives combination of terminal and variable. Without S→ACaB other products violate the rules for type- 3.

For type- 2, mentioned all products (w/o S→ACaB) aren’t valid because according to type- 2 restriction left hand portion contain only a variable. When we check for type- 1 then we see that all the products are valid because the left side portion length is equal to or less than the right side portion.

So, given products follow both type- 1 and type- 0.

**Question-2:**

The grammar with terminals {a, b, c, x}, non-terminals or values {S, A, B, C, X}. Then given products follow which type(s) of grammar?

S→Xa

X→a

X→aX

X→abc

X→ε (ε means null)

**Solution:**

Only when all products are valid for a type of grammar then we can say valid for that type of grammar.

First we should check from type-3 grammar but mentioned products violate the type-3 grammar restrictions cause for type-3 we know that is valid only when one variable gives only a terminal or when one variable gives the combination of terminal and variable. For X→abc break the condition cause here in the right portion remain 3 terminals.

For type- 2, mentioned all products are valid because according to type- 2 restriction left hand portion contain only a variable.

So, given products follow type- 2, type- 1 and type- 0.