Grammar

Grammar —Generates→ Language ←Recognizes— Automata.

Automata

A algorithm or program that automatically recognizes if a pattern or a particular string belongs to the language or not, by checking the grammar of the string.

An Automata is an abstract computing device or machine. There are different varities of such abstract machines (also called models of computation) which can be defined mathematically.

Definition of a Grammar

A - phrase structure grammar is represented by using 4 tuples.

1) V_N is a finite non-empty set whose elements are called variables.

2) Σ is a finite non-empty set whose elements are called terminals.

3) S is a special variable (an element of V_N) called the start symbol.

4) P is called productions or production rules or re-writing rules, which is a finite set whose

elements are $\alpha \rightarrow \beta$, where α and β are strings on $V_N \cup \Sigma$. 'α' has atleast one symbol from V_N.

* Variables or non-terminals are written using capital letters.
* Terminals are written using small letters.
* In production rule left-hand side must have ~~one~~ atleast one variable or non-terminal.
* The production rule in the form '$\alpha \rightarrow \beta$', where

$$\alpha \in (\Sigma \cup V_n)^* V_n (\Sigma \cup V_n)^*$$

$$\beta \in (\Sigma \cup V_n)^*$$

That means in production rule ~~Right~~ left hand side must have atleast one Variable.

Chomsky Hierarchy

Noam Chomsky introduce this in 1956.

It is the broad classification of the various types of grammar available.

Grammars are classified by the form of their Production.

Each category represents a class of languages that can be recognized by a different automaton.

The classes are nested, with type 0 being the largest and most general and type 3 being the smallest and most restricted.

Grammar	Language	Machine
Type 0	Recursive Enumerable Language	Turing Machine.
Type 1	Context Sensitive Language	Linear Bound Automata
Type 2	Context Free Language	Push Down Automata
Type 3	Regular Language	Finite Automata.

Type '0' Grammar

Type '0' Grammars are also known as 'Unrestricted Grammar' or Recursive Enumerable Grammar or Phase Structured Grammar.

It includes all formal grammars.

They generate exactly all languages that can be recognized by a TM.

These languages are also known as the Recursive Enumerable Language.

The productions can be in the form →

$$a \to b$$

where $a \in (\Sigma \cup V_N)^* V_N (\Sigma \cup V_N)^*$

and $b \in (\Sigma \cup V_N)^*$

Here L.H.S must contain atleast one Variable or non-terminal.

ex:- $S \to AaB$

$Bc \to acB$

Type 1 Grammar

Type '1' Grammar are also known as context Sensitive Grammar. It generates context-sensitive Language which can be recognized by a linear Bound Automata (LBA).

The production rule in the form →

$$a \to b$$

where $a \in (\Sigma \cup V_N)^* V_N (\Sigma \cup V_N)^*$

and $b \in (\Sigma \cup V_N)^+$

also $|a| \leq |b|$

In another world if production rule in the form →

$$aAb \to a\delta b$$

where $a, b \in (\Sigma \cup V_N)^*$

$A \in V_N$

and $\delta \in (\Sigma \cup V_N)^+$

The rule of the form $\boxed{S \to \in}$ is not allowed unless S is a start symbol and it does not occur on the right-hand side of any rule.

ex:- $AB \to CDB$

$B \to b$

$ABcd \to abcDBcd.$

$AB \to AbBC$

Type 2 Grammar

Type 2 Grammar are also known as Context-Free Grammar. It generates context free language which is accepted by Push Down Automata.

The production rule in the form →

$$a \rightarrow b$$

where $a \in V_N$

$|a| = 1$

$b \in (\Sigma \cup V_N)^*$

Context Free languages are the theoretical basis for the syntax of most programming language.

Ex:- $A \rightarrow aBc$

$A \rightarrow ab$

$S \rightarrow Aa$

Type 3 Grammar

Type 3 Grammar or Regular Grammar generates the regular language. Such a grammar restricts its rule to a single variable on the left-hand side and right-hand side consisting of a single terminal possibly followed or preceded but not both in the same grammar by a single Non-terminal.

$S \rightarrow \epsilon$ is also allowed here if S does not appear on the right hand side of any rule.

Regular language is accepted by Finite Automata.

Regular Grammar can be two types →

1) Left-linear Grammar

In this type of regular grammar, all the non-terminals on the right-hand side exist at the leftmost place.

$A \rightarrow Ba$
$A \rightarrow \in$
$A \rightarrow a$

where
$A, B \in V_N$
$|A| = |B| = 1$
$a \in \Sigma^*$

2) Right-linear Grammar

In this type of regular grammar, all the non-terminals on the right-hand side exist at the rightmost place.

$A \rightarrow aB$
$A \rightarrow \in$
$A \rightarrow a$

where
$A, B \in V_N$
$|A| = |B| = 1$
$a \in \Sigma^*$