**INTRODUCTION OF FLAT**

**FLAT**

* Formal language & automata theory
* Study of **computing machines** using mathematics.

**Alphabet**

* Non-empty & finite set of symbols.
* Denoted by **Σ**.
* Examples:

**Σ = {a.b.c…..z}**

**Σ = {0,1}**

**Σ = {1}**

**String**

* Finite **sequence** of symbols from **alphabet**.
* Denoted by **w**.
* For example:

**Σ = {0,1}**

**w = 01, 0010, 110**

**Length of String**

* Number of symbols involved in a **sequence**.
* Denoted by **|w|**.
* For example:

**Σ = {0,1}**

**w = 01, 0010, 110**

**|w| = 2, 4, 3**

**Empty String**

* An empty sequence (**|w| = 0**).
* Denoted by **€** (**w = €**).

**w.€ = w = €.w**

**Substring**

* If there exist **strings** in an **alphabet** & one of them is part of at least one other **string**, then that **string** is **substring**.
* Denoted by **u**.
* Every **string** is **substring** to itself.
* Empty string (**€**) is substring for everything.

**|u| <= |w|**

**Types of Substrings**

* Trivial substring
* Non-trivial substring

Trivial substring:-

* Also known as ***improper substring***.
* If there is a string **w**, then then the substring **w** itself & **€** are together called ***trivial substring***.

Non-trivial substring:-

* Also known as ***proper substring***.
* If there is a string **w**, then then any substring of **w** other than **w** itself & **€** are together called ***non-trivial substring***.

**Facts About Substring**

For **|w| = n**,

* No. of substrings **=** **Σn+1 = n(n-1)/2**
* No. of trivial strings = **2**
* No. of non-trivial substrings **= Σn-1**
* No. of non-empty substring **= Σn**
* No. of substrings of distinct length **= n**
* No. of strings of length **'n'** generated over alphabet **Σ = |Σ|n**

**Prefix and Suffix**

* **Prefix:** Sequence of starting symbol.
* **Suffix:** Sequence of ending symbol.
* For example, **w = TOC:**

**Prefix = €, T, TO, TOC**

**Suffix = TOC, OC, C, €**

**Power of an Alphabet**

* For example:

**Σ = {0,1}**

**Σ = {00,01,10,11}**

* **Positive closure:** Set of strings where **Σ+ = (|w| >= 1)**
* **Keen closure:** Set of strings where **Σ\* = (|w| >= 0)**

**Language**

* Collection of strings from **alphabet**.
* For example, if:

**Σ = {0,1}**

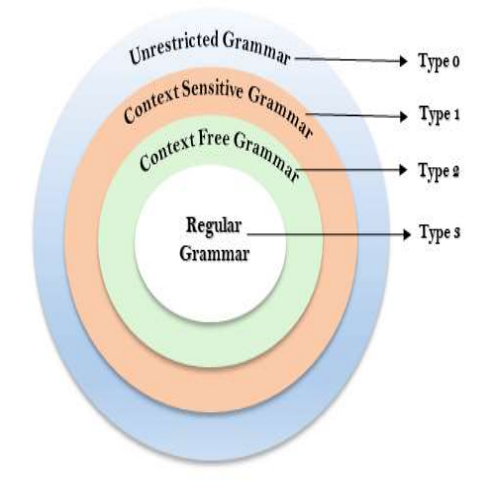
**L = {00,01,10,11}**

* **Σ\*** represents ***universal language***.

**Formal Language**

* When we put some **restriction** on the formation of **string**, we call it ***formal language***.

**Chomsky Classification of Formal Language**



* **Type 0 –** Recursive enumerable languages/ unrestricted grammar
* **Type 1 –** Context sensitive languages
* **Type 2 –** Context-free languages
* **Type 3 –** Regular languages

**Types of Languages**

* **Empty language:** It can also be an ***empty string***.
* **Non-empty language:** Contains at least **one** **string**.
* **Finite language:** Contains finite number of strings with **finite length**.
* **Infinite language:** Contains **infinite** number of strings but with finite length.

**Automata**

* **Mathematical system** which can represent formal language.

Types of automata:-

* ***Finite automata (FA)***
* ***Push down automata (PDA)***
* ***Linear bound automata (LBA)***
* ***Turing machine (TM)***

Expressive power:-

* ***Expressive power*** is the number of languages accepted by the automata.
* E(FA) = 1
* E(PDA) = 2
* E(LBA) = 3
* E(TM) = 4

**Grammar**

* Collection of rules that define **how to write** a string are called ***grammar***.
* **Grammar (G)** is collection of four tuples namely **V**, **T**, **P** and **S**.

**G = {V, T, P, S}**

**V = Non terminal symbols/ variables**

**P = Productions**

**T = Terminal symbols**

**S = Starting symbol**

**A 🡪 XYZ (r1)**

**X 🡪 a (r2)**

**Y 🡪 b (r3)**

**Z 🡪 c (r4)**

**V = {A, X, Y, Z}**

**P = {r1, r2, r3, r4}**

**T = {a, b, c}**

**S = {A}**

**Other Grammar Classifications**

* Other than the **4 types of grammars** we discussed, there are few more.
* **Recursive grammar:** When a variable is both on **left** & **right** side of a **production**.
* For example:

**S 🡪 aSb|$**

***\*Notice how S is on both sides\****

* **Non-recursive grammar:** Each element in production is on **either** side of the production, not both.

**S 🡪 ab|$**

**Derivation**

* Its **deriving** a string.
* **Derivation tree/ parse tree:** Geometrical representation of derivation.
* **Sentential:** Steps involved in deriving something.

