Elementary Concepts: -

Def 1: - A "symbol" is an abstract entity that we shall not define formally, just letters, and digits are examples of frequently used symbols.

Def 2: - An alphabet ∑ is a finite set of symbols (or characters). For example ∑={0, 1} is an alphabet with two symbols, ∑={a,b} is another alphabet with two symbols and the English alphabet is also an alphabet. In the context of strings, an alphabet is a finite set, whose elements are called symbols.

Def 3: - A string (also called a word) over an alphabet ∑ is a finite sequence of symbols, where each symbol is an element ∑ b, a, and abab are examples of strings over the alphabet {a, b}, and 0, 10, and 001 are examples of strings over the alphabet {0, 1}. A null string is a string with no symbols, usually denoted by epsilon ε.

Def 4: - The **length** of a string w, denoted by **|w|**, is the number of symbols contained in w. The empty string denoted by ε, is the string having length zero. Thus, |00100| = 5, |aab| = 3, | ε| = 0.

For example, if the alphabet is equal to {0, 1}, then, 10, 1000, 101, and ε are strings over ∑ , having lengths 2, 4, 3, and 0 respectively.

Def 5: - The "**concatenation**" notation to the concatenation (the dot .), x.y of two strings x and y consists of all characters of x followed by the characters of y as x.y = {v.s / v ϵ x, s ϵ y }.

For example: x= for, y= get, then x.y=forget>

Notes: -

1. x.y ≠ y.x
2. The concatenation is not commutative for every string.
3. The empty string is an identity for concatenation operation.

Def 6: - ***Prefix, suffix*** of string:

Let x, y two strings over ∑ then, a string **x** is called a **prefix** of **y** if a string z exists such as **y=xz**. Similarly, **x** is a proper **suffix** of **y** if there exists **z** such that **y=zx**.

For example: **w =abc**

Prefix → a, ab, abc

Suffix → c, bc, abc

Def 6: -

A **language** is the set of all strings of terminal symbols derivable from the alphabet.

For example, {a, ab, baa} is a language (over alphabet {a, b}) and {0, 111} is a language (over alphabet {0, 1}). Where,

L1 = Ø (the empty language).

L2 = {ε} (the language containing just the empty string, notice that L1 ≠ L2).

L3= {a, b, c}

L4= {ab, aab, aaab,….}, L4= $\{a^n b, n \geq 1\}$

Kinds of languages:

1. **Talking language**: (e.g.: English, French, Italy, …), it has an alphabet, for example, {a, b, c…z} from these alphabetic we make sentences that belong to the language.

Now we want to know if this sentence is true or false so we need **grammar**.

For example: Ali is a clever student. (It is a sentence in the English language).

2. **Programming language**: (e.g.: C++, Pascal, Java, …):

It has it has alphabet:={a, b, c,…z, A,B, C,…Z, ?, /, -, \, .}. From these alphabetic, we make sentences that belong to the programming language.

Now we want to know if this sentence is true or false so we need a **compiler** to make sure that syntax is true.

3. **Formal language**: (any language we want). It has strings from these strings we make sentences that belong to this formal language.

Now we want to know if this sentence is true or false so we need **rules.**

Example1:

Alphabet: ∑= {0, 1}.

Sentences: 0000001, 1010101.

Rules: **Accept any sentence starting with zero and refuse sentences that start with one.**

So we accept 0000001 as a sentence that satisfies the rules. And refuse: 1010101 as a sentence doesn't satisfy the rules.

Example2:

Alphabet: ∑= {a, b}.

Sentences: ababaabb, bababbabb

Rules: **Accept any sentence starting with a and refuse sentences that start with b.**

So we accept ababaabb as a sentence that satisfies the rules. And refuse: bababbabb as a sentence doesn't satisfy the rules.