Introduction of Theory of Computation

Automata theory (also known as **Theory Of Computation**) is a theoretical branch of Computer Science and Mathematics, which mainly deals with the logic of computation with respect to simple machines, referred to as automata.

Automata* enables scientists to understand how machines compute the functions and solve problems. The main motivation behind developing Automata Theory was to develop methods to describe and analyze the dynamic behavior of discrete systems.

Automata originated from the word “Automaton” which is closely related to “Automation”.

Basic Terminologies of Theory of Computation:

Now, let’s understand the basic terminologies, which are important and frequently used in the Theory of Computation.

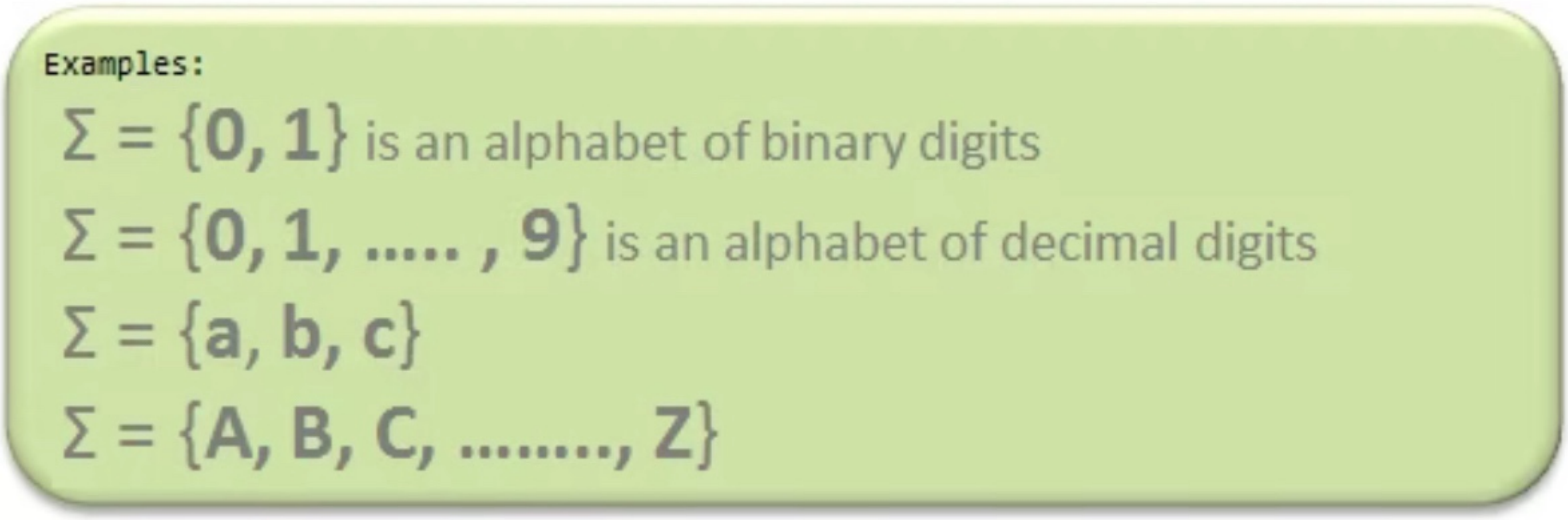
Symbol:

A symbol (often also called a **character**) is the smallest building block, which can be any alphabet, letter, or picture.

a, b, c, 0, 1, ...

Alphabets (Σ):

Alphabets are a set of symbols, which are always ***finite***.



Examples:

$\Sigma = \{0, 1\}$ is an alphabet of binary digits

$\Sigma = \{0, 1, \ldots\ldots, 9\}$ is an alphabet of decimal digits

$\Sigma = \{a, b, c\}$

$\Sigma = \{A, B, C, \ldots\ldots\ldots, Z\}$

String:

A string is a ***finite*** sequence of symbols from some alphabet. A string is generally denoted as ***w*** and the length of a string is denoted as ***IwI***.

```
Empty string is the string with

zero occurrence of symbols,

represented as ε.
```

```
Number of Strings (of length 2)

that can be generated over the alphabet {a, b}:

                      -   -

                      a   a

                      a   b

                      b   a

                      b   b


Length of String |w| = 2

Number of Strings = 4
```

Closure Representation in TOC:

L^+: It is a ***Positive Closure*** that represents a set of all strings except Null or ε-strings.

L^*: It is "**Kleene Closure**", that represents the occurrence of certain alphabets for given language alphabets from zero to the infinite number of times. In which ε-string is also included.

From the above two statements, it can be concluded that:

```
L∗ = εL⁺
```

```
Example:
(a) Regular expression for language accepting all
                                         R = g*
                               R={ε,g,gg,ggg,gggg,

(b) Regular Expression for language accepting all
                                         R = g⁺
                               R={g,gg,ggg,gggg,gg
```

Language:

A language is a *set of strings*, chosen from some Σ* or we can say- ‘A language is a subset of Σ* ‘. A language that can be formed over ‘ Σ ‘ can be **Finite** or **Infinite**.

```
Example of Finite Language:

          L1 = { set of string of 2 }

         L1 = { xy, yx, xx, yy }


Example of Infinite Language:

         L1 = { set of all strings starts with 'b'

         L1 = { babb, baa, ba, bbb, baab, .......
```

Relationship between grammar and language in Theory of Computation

A grammar is a set of production rules which are used to generate strings of a language. In this article, we have discussed how to find the language generated by a grammar and vice versa as well.

Language generated by a grammar –

Given a grammar G, its corresponding language L(G) represents the set of all strings generated from G. Consider the following grammar,

```
G: S-> aSb|ε
```

In this grammar, using S-> ε, we can generate ε. Therefore, ε is part of L(G). Similarly, using S=>aSb=>ab, ab is generated. Similarly, aabb can also be generated.

Grammar :

It is a finite set of formal rules for generating syntactically correct sentences or meaningful correct sentences.

Constitute Of Grammar :

Grammar is basically composed of two basic elements –

1. **Terminal Symbols –**
   Terminal symbols are those which are the components of the sentences generated using a grammar and are represented using small case letter like a, b, c etc.
2. **Non-Terminal Symbols –**
   Non-Terminal Symbols are those symbols which take part in the generation of the sentence but are not the component of the sentence. Non-Terminal Symbols are also called Auxiliary Symbols and Variables.

Formal Definition of Grammar :

Any Grammar can be represented by 4 tuples – <N, T, P, S>

- **N** – Finite Non-Empty Set of Non-Terminal Symbols.
- **T** – Finite Set of Terminal Symbols.
- **P** – Finite Non-Empty Set of Production Rules.
- **S** – Start Symbol (Symbol from where we start producing our sentences or strings).

Production Rules :

A production or production rule in computer science is a rewrite rule specifying a symbol substitution that can be recursively performed to generate new symbol sequences. It is of the form α-> β where α is a Non-Terminal Symbol which can be replaced by β which is a string of Terminal Symbols or Non-Terminal Symbols.

Example-1 :

Consider Grammar G1 = <N, T, P, S>

```
T = {a,b}    #Set of terminal symbols
P = {A->Aa,A->Ab,A->a,A->b,A-> ϵ}    #Set of all p
S = {A}    #Start Symbol
```

As the start symbol is S then we can produce Aa, Ab, a,b,ϵwhich can further produce strings where A can be replaced by the Strings mentioned in the production rules and hence this grammar can be used to produce strings of the form (a+b)*.