**Brief History:**

‐ Before 1930’s, no any computer were there and Alen Turing introduced an abstract machine that had all the capabilities of today’s computers. This conclusion applies to today’s real machines.

‐ Later in 1940’s and 1950’s, simple kinds of machines called finite automata were introduced by a number of researchers.

‐ In late 1950’s the linguist N. Chomsky begun the study of formal grammar which are closely related to abstract automata.

‐ In 1969 S. Cook extended Turing’s study of what could and what couldn’t be computed and classified the problem as:

o Decidable o Tractable/intractable.

**Basic concepts of Automata Theory:**

‐ The basic terms that pervade the theory of automata include “alphabets”, “strings”, “languages”, etc.

## Alphabets: - (Represented by ‘∑’)

Alphabet is a finite non-empty set of symbols. The symbols can be the letters such as {a, b, c}, bits {0, 1}, digits {0, 1, 2, 3… 9}. Common characters like $, #, etc.

{0,1} – Binary alphabets

{+, −, \*} – Special symbols

## Strings: - (Strings are denoted by lower case letters)

String is a finite sequence of symbols taken from some alphabet. E.g. 0110 is a string from binary alphabet, “automata” is a string over alphabet {a, b, c … z}.

**Empty String: -**

It is a string with zero occurrences of symbols. It is denoted by ‘ε’ (epsilon).

## Length of String: -

The length of a string w, denoted by | w |, is the number of positions for symbols in w. we have for every string s, length (s) ≥ 0.

| ε | = 0 as empty string have no symbols.

| 0110 | = 4

## Power of alphabet: -

The set of all strings of certain length k from an alphabet is the kth power of that alphabet. i.e. ∑k = {w / |w| = k}

If ∑ = {0, 1} then,

∑0 = {ε}

∑1 = {0, 1}

∑2 = {00, 01, 10, 11}

∑3 = {000, 001, 010, 011, 100, 101, 110, 111}

## Kleen Closure: -

The set of all the strings over an alphabet ∑ is called kleen closure of ∑ & is denoted by ∑\*. Thus, kleen closure is set of all the strings over alphabet ∑ with length 0 or more.

∴∑\* = ∑0 ∪ ∑1 ∪ ∑2 ∪ ∑3 ∪ ……………

E.g. A = {0}

A\* = {0n / n = 0, 1, 2, …}

## Positive Closure: -

The set of all the strings over an alphabet ∑, except the empty string is called positive closure and is denoted by ∑+.

∴∑+ = ∑1 ∪ ∑2 ∪ ∑3 ∪ ……………

## Language: -

A language L over an alphabet ∑ is subset of all the strings that can be formed out of ∑; i.e. a language is subset of kleen closure over an alphabet ∑; L ⊆ ∑\*. (Set of strings chosen

from ∑\* defines language). For example;

* Set of all strings over ∑ = {0, 1} with equal number of 0’s & 1’s.
  1. = {ε, 01, 0011, 000111, ………}
* φ is an empty language & is a language over any alphabet.
* {ε} is a language consisting of only empty string.
* Set of binary numbers whose value is a prime:
  1. = {10, 11, 101, 111, 1011, ……}

## Concatenation of Strings: -

Let *x* & *y* be strings then *xy* denotes concatenation of *x* & *y*, i.e. the string formed by making a copy of *x* & following it by a copy of *y*.

More precisely, if *x* is the string of *i* symbols as *x* = a1a2a3…a*i* & *y* is the string of *j* symbols as *y* = b1b2b3…b*j* then *xy* is the string of *i* + *j* symbols as *xy* = a1a2a3…a*i*b1b2b3…b*j*.

For example; *x* = 000 *y* = 111 *xy* = 000111 & *yx* = 111000

Note: ‘ε’ is identity for concatenation; i.e. for any *w*, ε*w* = *w*ε = *w*

## Suffix of a string: -

A string *s* is called a suffix of a string *w* if it is obtained by removing 0 or more leading symbols in *w*. For example; *w* = abcd *s* = bcd is suffix of *w*. *s* is proper suffix if *s* ≠ *w*.

## Prefix of a string: -

A string *s* is called a prefix of a string *w* if it is obtained by removing 0 or more trailing symbols of *w.* For example; *w* = abcd *s* = abc is prefix of *w*,

Here, *s* is proper suffix i.e. *s* is proper suffix if *s ≠ w*.

## Substring: -

A string *s* is called substring of a string *w* if it is obtained by removing 0 or more leading or trailing symbols in *w*. It is proper substring of *w* if *s ≠ w*.

If *s* is a string then *Substr (s, i, j)* is substring of *s* beginning at *i*th position & ending at *j*th position both inclusive.

## Problem: -

A problem is the question of deciding whether a given string is a member of some particular language.

In other words, if ∑ is an alphabet & L is a language over ∑, then problem is; ‐ Given a string w in ∑\*, decide whether or not *w* is in L.