Unit-1
Basic Foundations

Purpose of TOC → To develop formal mathematical models of computation that reflect real-world computers.

⊛. **Review of Set Theory, Logic, Functions, Proofs:** [less imp]

1. Sets: A set is a collection of well defined objects. For example: The set of odd positive integer less than 15 can be expressed by:

$S = \{1, 3, 5, 7, 9, 11, 13\}$.

or, $S = \{x \mid x \text{ is odd and } 0 < x < 15\}$

A set is finite if it contains finite number of elements in a set, otherwise infinite. The empty set has no element and is denoted by $\emptyset$.

Cardinality of set → It represents number of elements within a set.

Subset → A set A is subset of a set B if each element of A is also element of B and is denoted by $A \subseteq B$.

Unit set → A set containing only one element.

Universal set → A set of all entities in the current context.

Power set → The set of possible subsets from any set is known as power set. A set with n elements has 2^n subsets.

⇒ There are some set operations like Union, Intersection, Complement, difference etc. we all know about them.

2. Logic: Logic is the study of the form of valid inference and laws of truth. A valid inference is one where there is a specific relation of logical support between the assumptions of the inference and its conclusion. In ordinary discourse, inferences maybe signified by words such as therefore, thus, hence and so on.

Propositional logic → It is the way of joining or modifying propositions to form more complicated propositions as well as logical relationships. In propositional logic there are two types of sentences: simple sentences and compound sentences. Simple sentences express atomic propositions about the world. Compound sentences express logical relationships between the simpler sentences of which they are composed.

Propositions → Proposition is a declarative sentence that is either true or false but not both.

Example: $2+6=4$ (False), is a proposition
$2+3=5$ (True), is a proposition
$x>5$ (True and False based on value of x), is not a proposition.

Predicate → Any declarative statements involving variables often found in mathematical assertion and in computer programs, which are neither true nor false when the values of variables are not specified is called predicate.

For example: $5>9$, is not a predicate rather it is propositional statement because it is false.

But the statement "$x>4$" is predicate. The result is neither true nor false, it depends on the value of variable 'x'. The predicate "$x>4$" has two parts variable part (i.e, x) called subject and another relation part (i.e, >4). We can denote the statement "$x>4$" by $P(x)$. Once value is assigned to the propositional function then we can tell whether it is true or false. i.e, a proposition.

Quantifiers → Quantifiers are the tools that change the propositional function into a proposition. These are the word that refers to quantifiers such as "some" or "all" and indicates how frequently a certain statement is true. Construction of prepositions from the predicates using quantifiers is called quantifications.

There are two types of quantifiers: Universal and Existential.

<u>Universal quantifier</u> → The phrase "for all" denoted by $\forall$, is called universal quantifier.

<u>Existential quantifier</u> → The phrase "there exist", denoted by $\exists$, is called existential quantifier.

<u>3. Functions:</u> If A and B are two sets, a function f from A to B is a rule that assigns to each element x of A an element f(x) of B. For a function f from A to B, we call A the domain of f and B the co-domain of f.

$$x \longrightarrow \boxed{f(x)} \longrightarrow y$$

Input Function Output.

<u>Relations</u> → A binary relation on two sets A and B is a subset of $A \times B$. For example, if $A = \{1, 3, 9\}$, $B = \{x, y\}$, then $\{(1, x), (3, y), (9, x)\}$ is a binary relation on 2-sets.

A binary relation r is an equivalance relation if R satisfies:

→ R is reflexive i.e, for every x, $(x, x) \in R$.

→ R is symmetric i.e, for every x and y, $(x, y) \in R$ implies $(y, x) \in R$.

→ R is transitive i.e, for every x, y and z, $(x, y) \in R$ and $(y, z) \in R$ implies $(x, z) \in R$.

<u>Closures</u> → Closure is an important relationship among sets and is a general tool for dealing with sets and relationship of many kinds. Let R be a binary relation on a set A. Then the reflexive closure of R is a relation such that:

→ R' is reflexive (symmetric, transitive).

→ $R' \supseteq R$

→ If R'' is a reflexive relation containing R then $R \supseteq R$.

4. Methods of proofs / Proof Techniques: (Direct, Indirect, Contradiction)

These are the things exactly we read in discrete structure in 2nd sem. Not much more imp here so I'am leaving it. Now, important part of this chapter starts from below.

⊛. Theory of Computation (TOC):

It is a study of power and limits of computing having three interacting components: Automata Theory, Computability Theory and Complexity Theory.

Computability Theory
- What can be computed?
- Are there problems that no program can solve?

Complexity Theory
- What can be computed efficiently?
- Are there problems that no program can solve in a limited amount of time or space?

Automata Theory
- Study of abstract machine and their properties, providing a mathematical notation of "computer".
- Automata are abstract mathematical models of machines that perform computations on an input by moving through a series of states or configurations. If the computation of an automata reaches an accepting configuration it accepts that input.

Study of Automata
- For software designing and checking behaviour of digital circuits.
- For designing software for checking large body of text as a collection of web pages, patterns etc. like pattern recognization.
- Designing "lexical analyzer" of a compiler, that breaks input text into logical units called "tokens".

Abstract Model: An abstract model of computer system ~~(considered)~~ (considered either as hardware or software) constructed to allow a detailed and precise analysis of how the computer system works. Such a model usually consists of input, output and operations that can be performed and so can be thought of as a processor. E.g. an abstract machine that models a banking system can have operations like "deposit", "withdraw", "transfer" etc.

⊛. Basic Concepts of Automata Theory [Imp];

Alphabets → 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 ... 9}, Common characters like $, #, * etc.
For Example:

$$\Sigma = \{0,1\} \rightarrow \text{Binary alphabets}$$
$$\Sigma = \{+, -, *\} \rightarrow \text{Special Symbols.}$$

Strings → String is a finite sequence of symbols taken from some alphabet. E.g. 0110 is a string from binary alphabet, "computation" is a string from alphabet {a, b, c, ... z}.

Length of string → The length of string w, denoted by $|w|$, is the number of symbols in w.
Example: string w = computation has length 11.

Empty string → It is a string with zero occurrences of symbols. It is denoted by 'ε' (epsilon). The length of empty string is zero. i.e, $|\varepsilon| = 0$.

Power of Alphabet → The set of all strings of certain length k from an alphabet is the k^{th} power of that alphabet i.e, $\Sigma^k = \{w / |w| = k\}$
If $\Sigma = \{0,1\}$ then,

$\Sigma^0 = \{\varepsilon\}$ $\qquad$ $\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^1 = \{0,1\}$ $\qquad$ $\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

Kleen Closure:

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

$\therefore \Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

E.g. $A = \{0\}$

$A^* = \left\{ \frac{0^n}{n=0,1,2,\ldots} \right\}$

Positive Closure: The set of all the strings over an alphabet Σ except the empty string is called positive closure and is denoted by Σ^*.

$\therefore \Sigma^* = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$

Concatenation of Strings: If x and y are two strings over an alphabet, the concatenation of x and y is written xy and consists of the symbols of x followed by those of y.

For example:

$x = aaa$

$y = bbb$

$xy = aaabbb$

$yx = bbbaaa$

Note → Concatenating the empty string 'ε' with another string, the result is just the other string.

Suffix of a string: String is called suffix of a string w if it is obtained by removing zero or more leading symbols in w.

For example:

$w = abcd$

$s = bcd$ is suffix of w

s is proper suffix if $s \neq w$

Scanned by TapScanner

Prefix of a string: A string is called prefix of a string w if it is obtained by removing zero or more trailing symbols of w.

For example:

$w = abcd$

$s = abc$ is prefix of w.

s is proper ~~suffix of s~~ prefix if $s \neq w$.

Substring: A string s is called substring of a string w if it is obtained by removing zero or more leading or trailing symbols in w. It is proper substring if $s \neq w$.

Language: A language L over an alphabet Σ is subset of all the strings that can be formed out of Σ. i.e, a language is subset of kleen closure over an alphabet. Σ: $L \subseteq \Sigma^*$. (Set of strings choosen from Σ^* defines language).

For example:

→ Set of all strings over $\Sigma = \{0,1\}$ that ends with 1.

$L = \{1, 01, 11, 011, 0111, \ldots\}$

→ English language is a set of strings taken from the alphabet

$\Sigma = \{a, b, c, \ldots z, A, B, C, \ldots, Z\}$.

$L = \{dbms, TOC, Graphics\}$.

Empty language: A language is empty if it does not have any strings within it. ϕ is an empty language and is a language over any alphabet. It does not contain any string.

Membership in a language: Membership in a language defines association of some string with the language. A membership problem is the question of deciding whether a given string is a member of some particular language or not. i.e, whether the string belongs to the given language or not.

Example: Given, L = {DBMS, TOC, GRAPHICS}, then for W="DBMS", the membership of w is true in L. ~~and hence w&L~~ For W="HELLO", the membership of w is false in L. ~~and hence w~~