# Module 1

## Introduction to the theory of computation

Computer Science is the systematized body of knowledge concerning computation**.**Computer Science has two major components.First,the fundamental ideas and models underlying computing,which can used as tools in the design of computing systems. Theory provides the concepts and principles that helps us to understand the general nature of discipline. Field of Computer Science includes special topics from machine design to programming. To study the basic underlying principles we construct the abstract models of computers and computations. The four models of computing that we study are :Finite automata,Push down automata, Linear bounded automata and Turing machine.These models are studied as language recognizers because computation can be considered as string manipulation..These models embody the important features that are common to both hardware and software**.**

## Sets

A *set* is a collection of "things," called the *elements* or *members* of the set

For eg: L={a,b,c,d}. Here a$\varepsilon$ L, z $\not\varepsilon$ L

A set with only one element is called singleton set

A set with no element is called empty set denoted by $\Phi$.

## Methods to represent a set

### 1.Enumeration Method

In this method elements of a set are listed and enclosed them in braces.

For eg: A={a,b,c,……z}

### 2.Standard Method:

Frequently used sets are usually given symbols that are reserved for them.

For eg: N :set of natural nos {0,1,2……}

Z: set of integers {….-2,-1,0,1,2…..}

Q:set of rational nos

R:set of real nos

### 3.Set builder Notation

For eg: Q={x/y : x,y $\not\varepsilon$ Z ,y$\neq$ 0 }

The above set defines rational nos. x/y indicates a typical element of the set which is a fraction such that x,y $\varepsilon$ Z.

### 4.Finite set

A set is finite if it has finite number of elements. For eg: S={1,2,3}

### 5.Infinite set

Any set that is not finite is an infinite set.For eg: S={1,2,3……}

**Cardinality:** Let S be a finite set.The total number of different elements in S is called its cardinality and is denoted by **|A|**

For eg: Let S={1,2,3,4,3} then **|A|** =4.

- Let $S_1$ and $S_2$ be two sets.We can say that $S_1$ is a subset of $S_2$ ($S_1 \subseteq S_2$) If and only if every element of $S_1$ is an element of $S_2$.
- We can say that $S_1$ is equal to $S_2$ ($S_1 = S_2$) if and only if every element of $S_1$ is an element of $S_2$ and conversely every element of $S_2$ is an element of $S_1$.

ie $S_1 \subseteq S_2$ and $S_2 \subseteq S_1$

- Two sets are disjoint if they have no common elements.
- The collection of all subsets of a given set set S is itself a set called power set of S and is deneoted by $2^S$
- Universal set of any set includes all possible elements in the universe that has properties of the elements in the given set.Letter U is used to denote universal set.

**Basic Set Operations**

**1.Union:**

The *union* of sets A and B, written A U B, is a set that contains everything that is in A, or in B, or in both.

**2.Intersection**

The *intersection* of sets A and B, written A ∩B, is a set that contains exactly those elements that are in both A and B.

**3.Difference:**

The *set difference* of set A and set B, written A - B, is a set that contains everything that is in A but not in B.

**Laws of Set Theory**

Let $S_1$, $S_2$ and $S_3$ are three sets

1.Communicative laws:

$S_1 \cup S_2 = S_2 \cup S_1$

$S_1 \cap S_2 = S2 \cap S_1$

2.Associative laws :

$S_1 \cup (S_2 \cup S_3) = (S_1 \cup S_2) \cup S_3$

$S_1 \cap (S_2 \cap S_3) = (S_1 \cap S_2) \cap S_3$

3.Distributed laws :

$S_1 \cap (S_2 \cup S_3) = (S_1 \cap S_2) \cup (S_1 \cap S_3)$

$S_1 \cup (S_2 \cap S_3) = (S_1 \cup S_2) \cap (S_1 \cup S_3)$

4.Identity laws:

$S_1 \cup \Phi = \Phi \cup S_1 = S_1$

$S_1 \cap U = U \cap S_1 = S_1$

5.Idempotent laws :

$S_1 \cup S_1 = S_1$

$S_1 \cap S_1 = S_1$

**Cartesian Product**

The Cartesian product of two sets *X* and *Y* denoted *X* × *Y*, is the set of all possible ordered pairs whose first component is a member of *X* and whose second component is a member of *Y* .

$X \times Y = \{(x,y) / x \in X, y \in Y\}$

A binary relation on set A and B is a subset of A X B.

A function is a rule that assigns to elements of one set a unique element of another set.If 'f' denotes functionthen the first set is called the domain of 'f' and the second set is called its range .

$f: S_1 \rightarrow S_2$

It indicates that the domain of 'f' is a subset of $S_1$ and the range of f is a subset of $S_2$ .If the domain of 'f' is all the elemnts in $S_1$ then f is a total function .Otherwise 'f' is said to be a partial function.

**One to One function:**

A function ,f:A →B is one to one or injective if for any two distinct elements a,a' ε A, f(a) ≠ f(a'). ie Different elements in X has different images

**Onto or Surjective Function:**

A function ,f:A →B is onto or surjective if each element of B is the image under 'f' of some element of A

**Bijective Function:**

A function ,f:A →B is a bijection between A and B if it is both one to one and onto.

**Relexive relation**

A relation R $\subseteq$ A X A is reflexive if (a,a) ε R for each a ε A

**Symmetric relation**

A relation R $\subseteq$ A X A is symmetric if (b,a) ε R whenever (a,b) ε R

**Transitive relation**

A relation R $\subseteq$ A X A is transitive if whenever (a,b) ε R and (b,c) ε R then (a,c) ε R

**Equivalence relation**

A relation that is reflexive,symmetric and transitive is called an equivalence relation.

**Equinumerous Sets:**

We can say that two sets A and B are equinumerous if there is a bijection f: A →B.If there is a bijection fn f: A →B,then there is a bijection $f^1$: B→ A .Hence equinumerosity is a symmetric relation.

**Countably Infinite Set**:

A set is said to be countably infinite set if it is equinumerous with N.
To show that a set A is countably infinite we must exhibit a bijection f between A and N

**Countable Set**

A set is said to be Countable if it is finite or countably infinite

**Uncountable Set**

A set is said to be uncountable if it is not countable

**Computable and Non computable Functions:**

If we can compute the value of all elements 'x' that belongs to the domain set of the function then it is called computable function.

A function f defined over a set A is said to be computable if there exists an algorithm to find the value of f(x) ¥ x ε A

Afunction defined over a set A is said to be noncomputable if there exist atleast one element 'x' in A such that there is no algorithm to find f(x).

**Three fundamental proof techniques**:

**1.Principles of mathematical induction:**

The principles of mathamatical induction states that any set of natural nos containing zero and with the property that it contains n+1 whenever it contains all numbers upto end including 'n'( set of all natural nos")

We can use this induction method as follows :I f we want to show that property 'p' holds for all natural nos.Then to prove it following the steps

**Basic step**:
First show that property 'p' is true for 0.
**Induction Hypothesis**:
Assume property 'p' holds for 0,1,2….n
**Induction step:**
Using induction hypothesis show that 'p' is true for n+1.Then by induction principle 'p' is true for all natural nos.

**2.Pigeonhole principle**
If $S_1$ and $S_2$ are non empty sets and $| S_1| > | S_2|$ then there is no one to one function from $S_1$ to $S_2$ (ie there is no bijection).In other words, if we attempt to pair the element of $S_1$(the pigeons) with the element of $S_{2(}$ the pigeonhole) sooner or later we have to put more than one pigeon in a pigeonhole.

**3.Diagonalization principle**:
Let R be a binary relation on set A and let D, the diagonal set for R be {a: aε A and (a,b) not an element in R.For each a εR, let $R_a$ ={ b:bε A and (a,b) ε R} then D is distinct from each $R_a$
If A is a finite set then R can be picturised as a square array: the rows and coklumns are labeled with elements of A and there is a cross in the box with row labeled 'a' and column labeled 'b' if (a,b) ε R. The diagonal set D is the complement of sequences of boxes along the main diagonal.The sets $R_a$ corresponds to the rows in the array.The diagonalization principle states that the complement of the diagonal is different from each row.
Diagonalization principle holds for infinite sets because the diagonal set D is different from $R_a$ for any 'a'.It is used to check whether a given set is countable or not.

**Chomsky Classification of Languages**
Noum Chomsky, a founder of formal language theory provided the initial classification into 4 types.
Type 0(Unrestricted Grammars

Type 1(Context sensitive grammars)
Type 2(Context free grammars)
Type 3(Regular grammars)

This classification is based on the format of productions.

**Type 0 grammars**:

Unrestricted grammar is defined as G=(V,T,S,P) and P is the set of productions which is of the form $\alpha \rightarrow \beta$ where $\alpha$ ,$\beta$ are arbitrary strings of the grammar with $\alpha \neq \varepsilon$.

- Largest family of grammars
- Recognizer for type 0 language is Turing machines
- Language generated by this grammar are called recursively enumerable languages

**Type 1 grammars**:

It is also known as Context sensitive grammars.Here production rule is of the form $\alpha \rightarrow \beta$ where $| \beta | >= | \alpha |$

- Recognizer for this type of grammar is Linear Bounded automata.It is the modification of Turing machine which is non deterministic in nature.Here input is having $ at the left and right end.
- Language generated by this type of grammar is context sensitive language

**Type 2 grammars :**

Every production is of the form $\alpha \rightarrow \beta$ where $\alpha$ ε. V and $\beta$ is a string of terminals and non terminals

- Also known as context free grammar
- Recognizer for this type of grammar is Push down automats.
- Useful for scanning,parsing
- Language generated by this type of grammar is context free language

**Type 3 grammars:**

Here production rule is of the form a single non terminal on LHS and RHS consisting of a single terminal possibly succeeded or preceeded by a single NT.
2 types of regular grammars

- Right linear – production is of the form $A \rightarrow \gamma B$ or $A \rightarrow \gamma$

- Left linear - production is of the form $A \rightarrow B\gamma$ or $A \rightarrow \gamma$

➢ Recognizer for regular language is finite automata

## Partial function

- A function is not defined for some of its domain.
- Let $X_1, X_2, \ldots, X_n$ are sets. And X represents the Cartesian product $X = X_1 \times X_2 \times \ldots \times X_n$ . Partial function from X to Y, for each $x_1$ in $X_1$, $x_2$ in $X_2$,.. $x_n$ in $X_n$, there exist **at most one** y in Y for which $(x_1, x_2, .. x_n, y) \in f$

For eg: $X_1 = \{0,1,2\}$ $X_2 = \{0,1,2\}$ $Y = \{a,b,c,d\}$

*f* = { (0,0,b), (1,2,d), (1,1,b), (0,1,c), (2,0,c) }

Function *f* is partial, because it is not defined for (0,2), (1,0), (2,1) and (2,2)

## Total function

- A function which is defined for all inputs of its domain.
- Let $X_1, X_2, \ldots, X_n$ are sets. And X represents the Cartesian product $X = X_1 \times X_2 \times \ldots \times X_n$ . Partial function from X to Y, for each $x_1$ in $X_1$, $x_2$ in $X_2$,.. ….$x_n$ in $X_n$, there exist **exactly one** y in Y for which $(x_1, x_2, .. x_n, y) \in f$

For eg:

$X_1 = \{0,1,2\}$ $X_2 = \{0,1,2\}$ $Y = \{a,b,c,d\}$

*f* = { (0,0,b), (1,2,d), (1,1,b), (0,1,c), (2,0,c), (0,2,a), (1,0,b), (2,1,c), (2,2,d) }

Complex functions are built by composition, primitive recursive functions and minimization

## Base functions and Strategy Set

***i. Integer Set***

1. Zero Function

   ZERO(x) = 0

   e.g., ZERO(3) = 0

2. Successor Function

   SUCC(x) = x+1

   e.g., SUCC(3) = 4

3. Selector Function

   $SEL(n,k) = I_k^n (X_1, X_2, .. X_n) = X_k$ $(k \leq n)$

   e.g., Given $I_3^5$ (10,6,4,8,2) , SEL(5,2) = 6

## Primitive Recursive Functions

- Functions defined in terms of base functions and strategy sets

***i. Integer Set***

1. Addition
   SUM(0,x) = SEL(1,1)x
   SUM(m+1, 0) = SUCC (SEL(3,2) (m, SUM(m,x) , x))

2. Multiplication
   PROD(0,x) = ZERO (x)
   PROD(m+1,x) = SUM ( SEL(3,2) (m, PROD(m,x), x) ,
   SEL(3,3) (m, PROD(m,x), x) )

3. Predecessor
   PRED (0) = SEL(0,0)
   PRED (m+1) = SEL(2,1) (m, PRED(m))

4. Factorial
   FACT(0) = SECC(ZERO(x))
   FACT(n+1) = PROD(FACT(n), SECC(n))

5. Exponentiation
   EXP(x,0) = SUCC (ZERO(x))
   *EXP(x,y+1)* = *PROD(*EXP(x,y), SEL(2,1)(x,y))

6. Proper Substraction

$$x \dot{-} y = \begin{cases} x\text{-}y & \text{if } y \leq x \\ 0 & \text{otherwise} \end{cases}$$

$x \dot{-} 0 \quad = \text{Sel }(1,1)(x)$

$x \dot{-} (y+1) \quad = \text{PRED }( x \dot{-} y )$

7. Modulus

$| x\text{-}y | \quad = \text{SUM}((x \dot{-} y), (y \dot{-} x))$

8.

$\text{MIN}(x , y) \quad = x \dot{-} (x \dot{-} y)$

$\text{MAX}(x,y) \quad = y + (x \dot{-} y)$