Curriculum Overview

Unit-1 Introduction to formal language	**LH-5**
Unit-2 Finite Automata and Regular Language	**LH-10**
Unit-3 CFG and Pushdown Automata	**LH-10**
Unit-4 Turing Machine	**LH-10**
Unit-5 Undecidability and Computational Complexity	**LH-7**
Unit-6 Automata Theory and Compiler	**LH-3**

Chapterwise Marks Distribution

Unit-1	**6**
Unit-2	13
Unit-3	13
Unit-4	14
Unit-5	9
Unit-6	5
Total	60

Unit-1 Covers

- Set
- Relation
- Function
- Proof technique
- Alphabets
- Language
- Regular expression

What is Theory of Computation?

- Theory of Computation’ or ‘Theory of Automata’ is the core area of computer science and engineering; it is the branch that aims to attempts the deep understanding of computational processes by means of effectively solving the problems via mathematical models, tools, and techniques.
- This understanding is important for its applications that include various model of computation like algorithm, compiler and VLSI design, to the creation of intelligent technology, cognitive psychology, and philosophy.

What is Theory of Computation?

- This broad area of computation is divided into three major branches:
- Computational Complexity theory
- Computability theory
- Automata theory and language

What is Theory of Computation?

Complexity theory:

- To be solving the problems via computers the first question rises in every one mind that is, “What makes some problems computationally hard and other problems are computationally easy?”.
- In a informal way a problem is called “computationally easy”, if it is efficiently solvable. For example of “easy” problems are as follows;

What is Theory of Computation?

1. Sorting a sequence of, say, 100 numbers,
2. Searching for a name in a telephone directory, and
3. Computing the fastest way to drive from kalanki to koteshwor etc.

What is Theory of Computation?

On the other hand, a problem is called “computationally hard”, if it cannot be solved efficiently, or if we are unable to determine that the problem will solve efficiently.

- For examples of “computationally hard” problems are as follows;

1. Time table scheduling for all courses at Carleton,
2. Factoring a 300-digit integer into its prime factors, and
3. Computing a layout for chips in VLSI etc.

What is Theory of Computation?

Computability theory:

- According to this theory in 1930’s Kurt Godel, Alonzo Church, Alan Turing, Stephen Kleene and Emil Post introduced a computational theory, that theoretical model proposed in order to understand which functional mathematical problems solvable and unsolvable led to the development of real computers.
- Computational theory is also known as recursion theory which is the result of studded computable functions and Turing degrees.

What is Theory of Computation?

Automata theory:

It is the study of abstract mathematical machine and it deals with definitions and properties of different types of “computation models”. Examples of such computational models are:

Finite Automata: These are used in text processing, compilers, and hardware design.

Context-Free Grammars: These are used to define programming languages and in Artificial Intelligence.

Context-Sensitive Grammars: It is less general than unrestricted grammars used for compiler designing and in Artificial intelligence.

Turing Machines: These form a simple abstract model of a “real” computer, such as your PC at home

History of Theory of automata

Before 1930's: Alan Turing Studied an abstract machine that had all the capabilities of today's computers to solve problems.

1931's to 1950's: Simpler kinds of machines were used which we called 'Finite Automata'.

Late 1950's to 1960's: N. Chomsky began the study of formal 'grammars' that are not strictly belongs to the machines, but these grammars have closer relationships to abstracts automata.

Important reasons why study theory of computation?

The major reasons about the importance to study of theory of computation are listed below;

- The importance to study the theory of computation is to better understand the development of formal mathematical models of computation that reflect the real-world of computer.
- To achieve deep understanding about the mathematical properties of computer hardware and software.
- Mathematical definitions of the computation and the algorithm.
- To rectify the limitations of computers and answer what kind of problems can be computed?

sets

- A set can be described in a number of different ways, the simplest is to list up all of its members if that is possible.
- For example , {1,2,3} is the set of three numbers 1,2,3.
- Every object separated between them is the member of the set.
- A set can also be described by listing the properties that its member must satisfy.
- For example, $(x|x\ 1\leq x\leq 2)$and x is a real number.
- A third way to represent a set is to give a procedure to generate the members of the set.
- In this first basic element of the set are presented, second method is given to generate elements of the set from known element of the set.
- Third a statement is given that excludes undesirable elements from the set

1.1 equality of set

- Two sets are equal if and only if they have the same elements.
- More formally, for any sets A and B , A=B if and only if $\forall x\ [x \in A \leftrightarrow x \in B]$
- For example,
- {1,2,3}={3,2,1}, that is the order of elements does not matter and {1,2,3}={3,2,1,1} that is duplication do not make any difference for the sets.

1.2 subset

In set theory, a subset is denoted by the symbol ⊆ and read as 'is a subset of'.

- Using this symbol we can express subsets as follows:
- **A ⊆ B; which means Set A is a subset of Set B.**
- Example: Find all the subsets of set A = {1,2,3,4}
- Solution: Given, A = {1,2,3,4}
- Subsets =
- {}
- {1}, {2}, {3}, {4},
- {1,2}, {1,3}, {1,4}, {2,3},{2,4}, {3,4},
- {1,2,3}, {2,3,4}, {1,3,4}, {1,2,4}
- {1,2,3,4}

1.3 cardinality

- **The cardinality of a set:**
- The cardinality of a set is the total number of unique elements in a set.
- **Example:**

A={1,6,7,8,9}

The cadinality of a set A is n(A)=5

Hence cardinality represents the number of elements in set.

1.4 Empty set

- The empty set is the unique set having no elements such that its cardinality is 0.
- The **empty set is denoted by** the symbol “Φ” and “φ” or { }.
- (i) Consider set A = {x : 3 < x < 4, x is a whole number} and this set A is the empty set, since there is no whole number between 3 and 4.

1.4 Universal set

- A universal set is a set which contains all the elements or objects of other sets, including its own elements. It is usually denoted by the symbol ‘U’.
- Suppose Set A consists of all even numbers such that, A = {2, 4, 6, 8, 10, …} and set B consists of all odd numbers, such that, B = {1, 3, 5, 7, 9, …}. The universal set U consists of all natural numbers, such that, U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10,….}. Therefore, as we know, all the even and odd numbers are a part of natural numbers. Therefore, Set U has all the elements of Set A and Set B.

1.5 Power set

- In set theory, the power set (or power set) of a Set A is defined as the set of all subsets of the Set A including the Set itself and the null or empty set. It is denoted by P(A).
- **Example of Power Set**
- Let us say Set A = { a, b, c }
- Number of elements: 3
- Therefore, the subsets of the set are:
- { } which is the null or the empty set
- { a }
- { b }
- { c }
- { a, b }
- { b, c }
- { c, a }
- { a, b, c }

1.5 Power set

- **The power set P(A) = { { } , { a }, { b }, { c }, { a, b }, { b, c }, { c, a }, { a, b, c } }**
- Now, the Power Set has $2^3 = 8$ elements.

Set operation

- The set operations are performed on two or more sets to obtain a combination of elements as per the operation performed on them. In a set theory, there are three major types of operations performed on sets, such as:

1. Union of sets (∪)
2. Intersection of sets (∩)
3. Difference of sets (–)
4. Complement of sets(‘)

Set operation

Union of Sets

- If two sets A and B are given, then the union of A and B is equal to the set that contains all the elements present in set A and set B. This operation can be represented as;
- $A \cup B = \{x: x \in A \text{ or } x \in B\}$
- Where x is the elements present in both sets A and B.
- Example: If set A = {1,2,3,4} and B {6,7}
- Then, Union of sets, $A \cup B = \{1,2,3,4,6,7$

Set operation

Intersection of Sets

- If two sets A and B are given, then the intersection of A and B is the subset of universal set U, which consist of elements common to both A and B. It is denoted by the symbol '∩'. This operation is represented by:
- **$A \cap B = \{x : x \in A \text{ and } x \in B\}$**
- Where x is the common element of both sets A and B.
- The intersection of sets A and B, can also be interpreted as:
- **$A \cap B = n(A) + n(B) - n(A \cup B)$**

Set operation

Difference of Sets

- If there are two sets A and B, then the difference of two sets A and B is equal to the set which consists of elements present in A but not in B. It is represented by A-B.
- Example: If A = {1,2,3,4,5,6,7} and B = {6,7} are two sets.
- Then, the difference of set A and set B is given by;
- A – B = {1,2,3,4,5}

Set operation

- **Complement of Set**
- If U is a universal set and X is any subset of U then the complement of X is the set of all elements of the set U apart from the elements of X.
- $X' = \{a : a \in U \text{ and } a \notin A\}$
- Example
  U ={6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25}

  A={9,16,25}

Therefore,

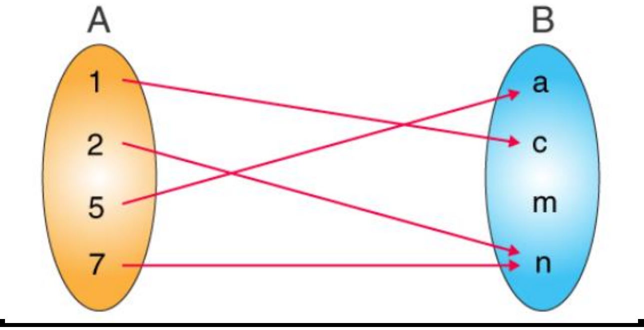
- A′ = {6,7,8,10,11,12,13,14,15,17,18,19,20,21,22,23,24}

Set operation

- **Commutative Property**
- $A \cup B = B \cup A$
- $A \cap B = B \cap A$
- **Associative Property**
- $A \cup (B \cup C) = (A \cup B) \cup C$
- $A \cap (B \cap C) = (A \cap B) \cap C$
- **Distributive Property**
- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
- $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

Relation

- **Relations and its types** concepts are one of the important topics of set theory.
- Sets, relations and functions all three are interlinked topics.
- Sets denote the collection of unordered elements whereas relations and functions define the operations performed on sets.
- The relations define the connection between the two given sets.



Relation

This mapping depicts a relation from set A into set B.

A relation from A to B is a subset of A x B.

- The ordered pairs are (1,c),(2,n),(5,a),(7,n). For defining a relation, we use the notation where,
- set {1, 2, 5, 7} represents the domain.
- set {a, c, n} represents the range.

Types of Relation

- **Empty Relation**
- An empty relation (or void relation) is one in which there is no relation between any elements of a set. For example, if set A = {1, 2, 3} then, one of the void relations can be R = {x, y} where, |x – y| = 8. For empty relation,
- **R = φ ⊂ A × A**
- **Universal Relation**
- A universal (or full relation) is a type of relation in which every element of a set is related to each other. Consider set A = {a, b, c}. Now one of the universal relations will be R = {x, y} where, |x – y| ≥ 0. For universal relation,
- **R = A × A**

Types of Relation

- **Identity Relation**
- In an identity relation, every element of a set is related to itself only. For example, in a set A = {a, b, c}, the identity relation will be I = {a, a}, {b, b}, {c, c}. For identity relation,
- **I = {(a, a), a ∈ A}**
- **Inverse Relation**
- Inverse relation is seen when a set has elements which are inverse pairs of another set. For example if set A = {(a, b), (c, d)}, then inverse relation will be R^{-1} = {(b, a), (d, c)}. So, for an inverse relation,
- **R^{-1} = {(b, a): (a, b) ∈ R}**

Types of Relation

- **Reflexive Relation**
- In a reflexive relation, every element maps to itself. For example, consider a set A = {1, 2,}. Now an example of reflexive relation will be R = {(1, 1), (2, 2), (1, 2), (2, 1)}. The reflexive relation is given by-
- **(a, a) ∈ R**
- **Symmetric Relation**
- In a symmetric relation, if a=b is true then b=a is also true. In other words, a relation R is symmetric only if (b, a) ∈ R is true when (a,b) ∈ R. An example of symmetric relation will be R = {(1, 2), (2, 1)} for a set A = {1, 2}. So, for a symmetric relation,
- **aRb ⇒ bRa, ∀ a, b ∈ A**

Types of Relation

- **Transitive Relation**
- For transitive relation, if $(x, y) \in R$, $(y, z) \in R$, then $(x, z) \in R$. For a transitive relation,
- **aRb and $bRc \Rightarrow aRc \ \forall\ a, b, c \in A$**
- **Equivalence Relation**
- If a relation is reflexive, symmetric and transitive at the same time, it is known as an equivalence relation.

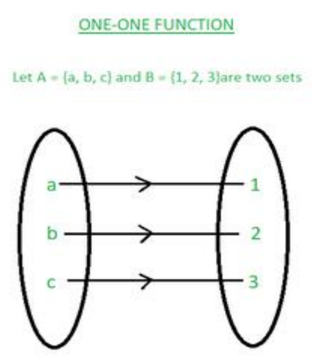
function

- Functions are an important part of discrete mathematics.
- A function assigns exactly one element of a set to each element of the other set.
- Functions are the rules that assign one input to one output. The function can be represented as f: A ⇢ B.
- A is called the *domain of the function* and B is called the *codomain function*.

function

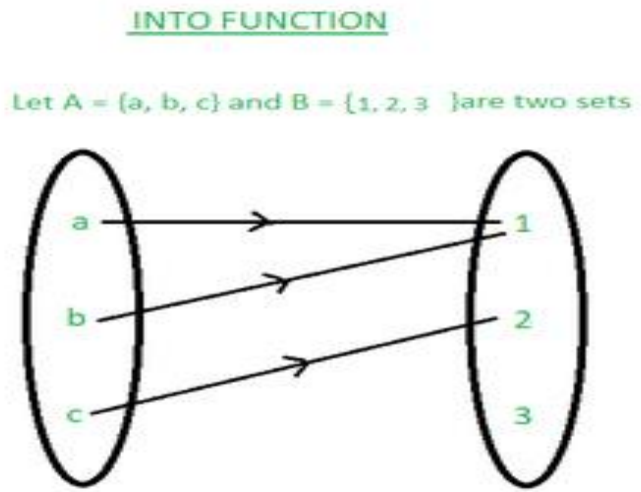
Types of function:

- **One-One function (or Injective Function):**
- A function in which one element of the domain is connected to one element of the codomain.
- A function f: A ⇢ B is said to be a one-one (injective) function if different elements of A have different images in B.
- **f: A ⇢ B is one-one**
- **$\Rightarrow$ $a \neq b \Rightarrow f(a) \neq f(b)$ for all a, b $\in$ A**
- **$\Rightarrow$ $f(a) = f(b) \Rightarrow a = b$ for all a, b $\in$ A**



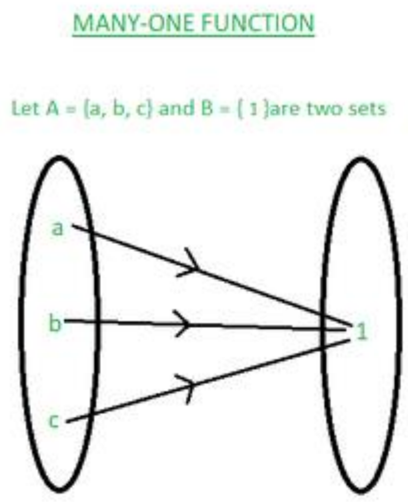
function

- **Into Function:**
- A function f: A ⇢ B is said to be an into a function if there exists an element in B with no pre-image in A.
- A function f: A⇢ B is into function when it is not onto.



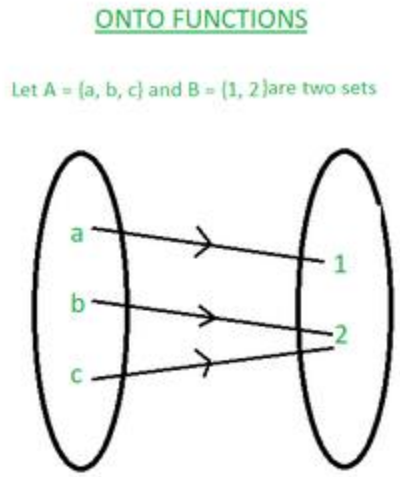
function

- **Many-One function:**
- A function f: A ⇢ B is said to be a many-one function if two or more elements of set A have the same image in B.
- A function f: A ⇢ B is a many-one function if it is not a one-one function.
- **f: A ⇢ B is many-one**
- **⇒ a ≠ b but f(a) = f(b)   for all a, b ∈ A**



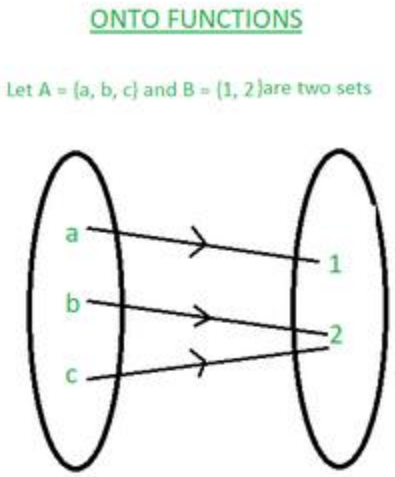
function

- **Onto function(or Surjective Function):**
- A function f: A -> B is said to be onto (surjective) function if every element of B is an image of some element of A i.e. f(A) = B or range of f is the codomain of f.
- **f: A ⇢ B is onto if for each b∈ B, there exists a∈ A such that f(a) = b.**



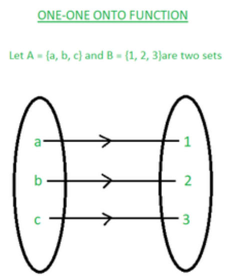
function

- **Onto function(or Surjective Function):**
- A function f: A -> B is said to be onto (surjective) function if every element of B is an image of some element of A i.e. f(A) = B or range of f is the codomain of f.
- **f: A ⇢ B is onto if for each b∈ B, there exists a∈ A such that f(a) = b.**



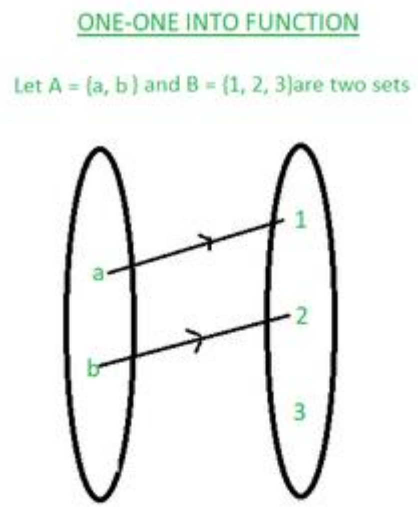
function

- **One-One Correspondent function(or Bijective Function or One-One Onto Function):**
- A function which is both one-one and onto (both injective and surjective) is called one-one correspondent(bijective) function.
- **f : A ⇢ B is one-one correspondent (bijective) if:**
- **one-one i.e. f(a) = f(b) ⇒ a = b for all a, b ∈ A**
- **onto i.e. for each b ∈ B, there exists a ∈ A such that f(a) = b.**



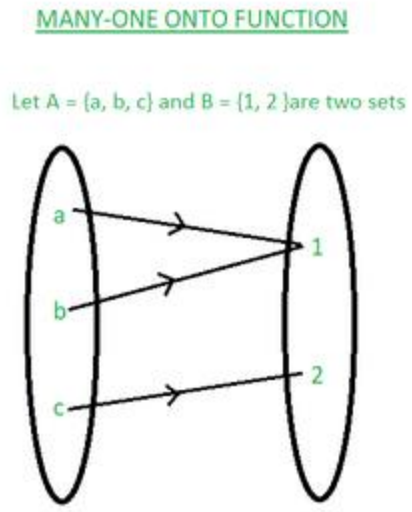
function

- **One-One Into function:**
- A function that is both one-one and into is called one-one into function.



function

- **Many-one onto function:**
- A function that is both many-one and onto is called many-one onto function.



function

- **Many-one into a function:**
- A function that is both many-one and into is called many-one into function.