Theory of Computation (CT-203)

Course Instructor
ANUJ GHIMIRE

Course Overview and Objective

About TOC

- It is one of the most fundamental course of Computer science.
- Will help to understand how people thought about the computer Science and Automation.
- It is mainly about what type of problem can really compute mechanically, how fast and how much of space does it take to do so.
- This course is about the fundamental capabilities and limitations of computers.

About TOC

- Nowadays, the Theory of Computation can be divided into the following three areas:
  - ***Automata Theory***
    - Deals with definitions and properties of different types of "computation models" (FA, PDA, Turing Machines).
  - ***Computability Theory***
    - Tries to classify problems as being solvable or unsolvable.
  - ***Complexity Theory***
    - Classifies problems according to their degree of "difficulty". Give a rigorous proof that problems that seem to be "hard" are really "hard".

Objectives of TOC

- **To provide basic understanding of theory to:**
  - *Build formal mathematical models of computation.*
  - *Analyse the inherent capabilities and limitations of these models.*
- **Course Goals:**
  - *Simple practical tools you can use in later courses, projects, etc. The course will provide you with tools to model complicated systems and analyse them.*
  - *Inherent limits of computers: problems that no computer can solve.*
  - *Better fluency with formal mathematics (closely related to skill at debugging programs).*

Applications of TOC

- Design of Compiler
- Pattern Matching
- Design of the Embedded System (Traffic Light System, Attendance through the Bio-Metrics).
- Natural Language Processing
- Vending Machines (ATM, Token Generation).

Syllabus

THEORY OF COMPUTATION
CT 203

Lecture : 3 **Year : II**
Tutorial : 1 **Part : I**

Course Objectives: To provide basic understanding of theory of automata, formal languages, computational models and computational complexity.

1. **Introduction to Formal Language, Logic and Proof (7 hours)**
   - 1.1. Brief Review of Set Theory, Function and Relation.
   - 1.2. Propositional Logic, Expressing Statements in Propositional Logic, Rules of Inference and Proofs in Propositional Logic, Introduction to Predicate Logic.
   - 1.3. Proofs, Principle of Mathematical Induction, Diagonalization Principle, Pigeonhole Principle
   - 1.4. Alphabet and Language
   - 1.5. Operations on Languages: Union, Concatenation, Kleene Star
2. **Finite Automata and Regular Language (10 hours)**
   - 2.1. Introduction to Finite Automata, Finite State Machine
   - 2.2. Deterministic Finite Automata (DFA), Representation of DFA, Language of DFA, Design of DFA
   - 2.3. Non Deterministic Finite Automata (NFA), Equivalence of DFA and NFA
   - 2.4. Finite Automata with Epsilon Transition (ε - NFA), Equivalence of NFA and ε – NFA, Equivalence of DFA and ε – NFA
   - 2.5. Regular Expressions and Regular Languages
   - 2.6. Equivalence of Regular Expression and Finite Automata
   - 2.7. Closure Properties of Regular Languages
   - 2.8. Pumping Lemma for Regular Languages
   - 2.9. Decision Algorithm for Regular Language

Syllabus

3. **Context Free Grammar and Pushdown Automata (10 hours)**
   - 3.1. Introduction to Context Free Grammar (CFG), Component of CFG, Context Free Language (CFL)
   - 3.2. Types of Derivations, Parse tree and its construction, Ambiguity
   - 3.3. Simplification of CFG, Normal Forms, Chomsky Normal Form (CNF), Greibach Normal Form (GNF), Backus-Naur Form (BNF)
   - 3.4. Closure Properties of Context Free Languages
   - 3.5. Pumping Lemma for Context Free Languages
   - 3.6. Decision Algorithm for Context Free Language
   - 3.7. Introduction to Push Down Automata (PDA), Representation of PDA, Operations of PDA, Move of a PDA, Instantaneous Description for PDA
   - 3.8. Language of PDA, Equivalence of CFL and PDA, Conversion of CFG to PDA
   - 3.9. Context Sensitive Grammar
4. **Turing Machine (10 hours)**
   - 4.1. Introduction to Turing Machine (TM), Representation of TM, Move of a TM, Instantaneous Description for TM
   - 4.2. Computing with Turing Machine

Syllabus

4.3. Variants of Turing Machine
4.4. Unrestricted Grammar, Chomsky Hierarchy of Grammar
4.5. Recursive Function Theory

5. Undecidability and Computational Complexity (5 hours)

5.1. Church Turing Thesis
5.2. Universal Turing Machine, Encoding of Turing Machine
5.3. Undecidable problem about Turing Machines, Halting Problems and its Implications
5.4. Computational Complexity, Time and Space complexity of a Turing Machine
5.5. Complexity Classes Class P, Class NP, NP-complete problems

6. Automata Theory and Compiler (3 hours)

6.1. Basic concept of Compiler, Role of Lexical Analyzer, Lexical Analysis with Deterministic Finite Automata
6.2. Parser and Context free Grammar, Top Down Parsing, Bottom Up Parsing, LR Parsing

References

1. H. R. Lewis, C. H. Papadimitriou, "Elements of theory of computation", Pearson Education.
2. Michael Sipser, "Introduction to the Theory of Computation", Thomson Course Technology.
3. Kenth Rosen, "Discrete Mathematical Structures with Applications to Computer Science", WCB/ McGraw Hill
4. Compilers Principles, Techniques, and Tools, Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman; Pearson Education

Syllabus

Evaluation Scheme:

There will be questions covering all the chapters in the syllabus. The evaluation scheme for the question will be as indicated in the table below:

Chapter	Hours	Marks Distribution*
1	7	9
2	10	13
3	10	13
4	10	14
5	5	6
6	3	5
Total	45	60

*There may be minor deviation in marks distribution

Evaluation Criteria

Let's Discuss !!!!

Evaluation Process

- Attendance
- Assignments
- Term Examinations
- MCQ
- Presentations or Report Submissions

DISCLAIMER

- *This document does not claim any originality and cannot be used as a substitute for prescribed textbooks.*
- *The information presented here is merely a collection from various sources as well as freely available material from internet and textbooks.*
- *The ownership of the information lies with the respective authors or institutions.*

Chapter 1: Introduction

1.1: Brief Review of Set Theory, Function and Relation

Set Theory

- A ***set*** is any well defined collection of distinct objects, called as elements or members of the set.
- Some examples of a set are:
  - All vowel alphabets
  - All district of Nepal
  - A collection of coins
- There is no restriction on the number of elements allowed in a set; there may be an infinite number, a finite number or even no elements at all.

Set Theory

- In general capital letters ***A, B, S, W, Z,*** .... are used to denote sets while the small letter ***a, b, c, x, y, z,..*** are used to denote the members of the sets unless other wise stated.
- If 'a' is an element of a set ***A*** we write this as $a \in A$ and read as "a belongs to the set ***A***". Again if 'a' is not an element of the set A we write this as $a \notin A$ and read as "a does not belong to the set A."
- If A be the set of natural numbers then it is written as:

$$A=\{1,2,3,4,\ldots\ldots\ldots\ldots\ldots\}$$

Representation of a Set

- A set may be specified by the following methods:
  - **Description Method**
    - *In this method a set is specified by a verbal description.*
    - *For example, the set S of numbers 1, 2 and 3 is designated as:*
      S= the set of positive integers less than 4.
  - **Tabulation Method**
    - *In this method a set is specified by listing all elements in a set. Thus, we can write the set S of numbers 1, 2 and 3 as:*

      S={1,2,3}
    - *Note that each of the sets {1, 2, 3}, {2, 3, 1} and {3, 1, 2} are the same.*

Representation of a Set

- **Rule Method or Set-builder method**
  - *Here a set is specified by stating a characteristic property common to all elements in the set.*

    $$S=\{x \rightarrow x \text{ is an integer and } 1 \leq x \leq 3\}$$

  - *This is read as the set of all elements x such that x is a positive integer less than 4. The vertical bar denotes 'such that'.*
  - *This method is suitable when the set has a larger number of elements. For example, if we take all men in Kathmandu owing Pulsar Bike it will be inconvenient to write the names of all persons within braces. But we can write this set briefly as:*
  - *S={x: x is a man in Kathmandu who owns Pulsar Bike}*

Special Set

- We refer to specific sets of numbers so often that we give them ***special names***.
- These sets, and their corresponding symbols, will be referenced throughout this course.
- ***Natural Numbers:***
  - We define the Natural Numbers to be:
    $$N = \{ 1, 2, 3, \ldots\ldots\ldots \}$$
  - Note that the natural numbers are "closed" under addition and multiplication.

Special Set

- ***Integers:***
  - We define the Integers to be:

$$\boldsymbol{Z} = \{\ldots, -2, -1, 0, 1, 2, 3, \ldots\}$$

  - Note that **Z** is "closed" under addition, subtraction, and multiplication.
- ***Rational Numbers:***
  - We define the Rationals to be:

$$\boldsymbol{Q} = \{\ p/q \mid p,q \in Z \text{ and } q \neq 0\}$$

  - Note that **Q** is "closed" under addition, subtraction, multiplication, and non-zero division.

Special Set

- ***Irrational Numbers:***
  - **I** = {all infinite, nonrepeating decimals}
  - Obviously, irrational numbers are impossible to write down exactly.
  - We use symbols to represent special values such as π, e, and $\sqrt{2}$.
  - The Irrationals are *not* closed under addition, subtraction or multiplication.

Special Set

- ***Real Numbers:***
  - **R** = {all decimal expansions}
  - The Real Numbers are created by adjoining the Rational with the Irrationals.
  - The Reals are closed under *all* operations.
- ***Complex Numbers:***
  - Real Number fall short when solving simple polynomial equations like $x^2 + 1 = 0$.
  - The Complex Numbers patch this hole.
  - $\mathbf{C} = \{a + bi \mid a,b \in \mathbf{R} \text{ and } i = \sqrt{(-1)}\}$

Singleton and Empty Set

- A set may have only one element; it is then called a singleton or unit set.
  - *For example, {1} is the set with 1 as its only element; thus {1} and 1 are quite different.*
- Empty set is a set with no element at all and is denoted by ϕ.
  - *P={x: x is the male students of Padma Kanya Campus}*
- Note that the set {0} is not an empty set since it contains zero as its element.
- Any set other than the empty set is said to be non-empty.

Finite and Infinite Set

- A set consisting of finite number of elements is called finite set.
  - The set of days in a week is a finite set.
- A set consisting of infinite number of elements is called infinite set.
  - A set of all odd numbers is an infinite set.
  - Thus, S= {1, 3, 5,.....} is an infinite set.

Disjoint and Overlapping Set

- Two or more sets are said to be disjoints, if there are no common elements among.
  - A={a, b, c}
  - B = {e, f, g}, here A and B are disjoint sets.
- Two or more sets overlapping set if there area at least one common element among.
  - A={a, b, c}
  - B = {c, d, e}, here A and B are overlapping sets.

Universal Set

- The set of all the entities in the current context is called the universal set, or simply the universe.
- In other word it is a set that has all the elements associated with a given set, without any repetition and is denoted by **U**.
- The context may be a integers, for example, where the Universal set is limited to the particular entities under its consideration. Also, it may be any arbitrary problem, where we clearly know where it is applied.

Subset

- A set that consists of some or all elements of another set is called subset of the set.
- The set A is subset of the set B if and only if each elements of A is also an element of B.
- In symbols we write $A \subseteq B$ (and read as 'A is subset of B) if and only if $x \in A$ implies that $x \in B$
- Some examples of a subset are as follows:
  - If A{1, 2} and B = {1, 2, 3} then, $A \subseteq B$.
  - Every set A is a subset of itself that is $A \subseteq A$.
  - Null set ϕ is a subset of any set S.
  - If A{a, b, c} and B {b, a, c}, then $A \subseteq$ Band $B \subseteq A$. Then A and B are called equal sets.

Power Set

- The possible subsets in a set {a} will be ϕ and {a}. Hence, the number of subsets that can be formed out of a set consisting of one element is $2^1 = 2$.
- Similarly, the possible subsets in a set {a, b} will be 4, i.e. ϕ, {a}, {b} and {a, b} which are $4 = 2^2$ in number.
- If we take a set {a, b, c} with the three elements a, b and c then its possible subsets will be $8 = 2^3$ in number.

Power Set

- Proceeding in this manner, we conclude by induction that a set with n elements has 2^n subsets. This includes the null set and the given set.
- So, the set of possible subsets from any set is known as power set of that set.
  - Let A={a, b, c} be any set then power set of A is
  - P(A)=(ϕ, {a}, {b}, {c}, {a, b}, {b, c}, {c, a} and {a, b, c})

Countably Infinite Set

- A set is said to be countably infinite set if its elements can be put one-one correspondence with the set of natural numbers. For example,
  - Z = {x: x is a element of integers}

    = {……-3, -2, -1, 0, 1, 2, 3, …..}
- Here, we can't never count the cardinality of sets but if we arrange elements is such a way that
  - Z={0, -1, 1, -2, 2, -3, 3, …}
  - And then if we are asked to count number of elements up to -3, we can do that. So. the set of integers is the countably infinite set.

Uncountable Set

- A set is uncountable if it contains so many elements that they can't be put one to one correspondence with the set of natural numbers.
- In other words, it is opposite to that of countably infinite set. For example,
  - The set of real numbers in {0, 1}.

Ordered Pair and Cartesian Product

- The pair in which first element belongs to first set and second element belongs to second set is called ordered pairs. For ordered pairs (a, b), $a \in A$ and $b \in B$.
- The Cartesian product of two sets A and B denoted by $A \times B$ is the set of all ordered pairs (a, b) with $a \in A$ and $b \in B$. Note that $A \times B \neq B \times A$
  - i.e. $A \times B=\{(a, b): a \in A \text{ and } b \in B\}$
  - $B \times A=\{(c, d): c \in B \text{ and } d \in A\}$
  - For example,
  - Let A(a, b) and B= (c, d) be the two sets

Then, $A \times B=\{(a, c), (a,d), (b, c), (b, d)\}$

$B \times A=\{(c, a), (c, b), (d, a), (d, b)\}$

$\boldsymbol{A \times B \neq B \times A}$

Ordered Pair and Cartesian Product

- Example_2

The Cartesian product of $\{1,3,9\} \times \{b,c,d\}$ is:

$$\{(1,b),(1,c),(1,d),\ (3,b),(3,c),(3,d),(9,b),(9,c),(9,d)\}$$

Set Operations

- ***Union:***
  - For any two sets A and B, the union of A and B, written as $A \cup B$, is the set of all elements which are members of the set A or the set B or both.
  - Symbolically, it is written as

    $$A \cup B=\{x|\ (x \in A) \text{ or } (x \in B)\}$$

  - From the definition, it follows that
    - $A \cup B=B \cup A$
    - $A \cup \phi = A$ and
    - $A \cup A=A$

Set Operations

- ***Intersection:***
  - The intersection of any two sets A and B, written as; $A \cap B$, is the set consisting of all the elements which belong to both A and B. Symbolically,
  - $A \cap B = \{x | (x \in A) \text{ and } (x \in B)\}$
  - From the above definition it follows that for any sets A and B
    - $A \cap B = B \cap A$
    - $A \cap A = A$
    - and $A \cap \phi = \phi$

Set Operations

- ***Difference:***
  - Let A and B be two sets and each set is the subset of a universal set U.
  - Then, A difference B denoted by ***A-B***, is the set of all those elements which belong to A but not B.
  - In symbols, we write this as:

$$A-B=\{x \mid x \in A \text{ and } x \notin B\} \text{Also,}$$

$$B-A=\{x \mid x \in B \text{ and } x \notin A\}$$

Set Operations

- ***Complement of Set:***
  - Let A be the subset of a universal set U. Then the complement of A with respect to U is the set of all those elements of U which do not belong to A and is denoted by $\bar{A}$ or A' or A^c.
  - In symbols, we write this as :

$$A^c = \{x \mid x \in U \text{ and } x \notin A\}.$$

Set Operations

- ***Symmetric Difference:***
  - Symmetric difference of two sets A and B is the set of all elements which are in either of the sets and not in their intersection.
  - The symmetric difference is denoted by:

$$A \Delta B = (A-B) \cup (B-A).$$

Properties of Set Operations

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

Properties of Set Operations

- ***DeMorgan's Law:***
  - $(A \cup B)^c = A^c \cap B^c$
  - $(A \cap B)^c = A^c \cup B^c$
- ***Idempotent Property:***
  - $A \cup A = A$
  - $A \cap A = A$

Relation

- A relation on sets S and T is a set of ordered pairs (s, t), where:
  - $s \in S$ *(s is a member of S)*
  - $t \in T$
  - *S and T need not be different.*
  - *The set of all first elements is the "domain" of the relation, and*
  - *The set of all second elements is the "range" of the relation.*

Relation

- Relation between sets is the every possible subset of Cartesian product between them.
- Generally, it is denoted by ***R***.
- Suppose S is the set {a, b, c, d, e} and set T is {w, x, y, z}. Then one of the relation on S and T can be:
  - R = {(a, y), (c, w), (c, z), (d, y)}
- ***Binary Relation:*** If relation is formed from subsets of Cartesian product of two sets then it is binary relation.

Types of Relation

- ***Reflexive Relation:***
  - A relation $R \subseteq A \times A$ is reflexive if $(a, a) \in R$ for each $a \in A$.
  - The directed graph representing a reflexive relation has a loop from each node to itself.
    - Let A= { 1, 2, 3} then
    - R= { (1, 1), (2, 2) , (3, 3) } is a reflexive relation defined on set A.

Types of Relation

- ***Symmetric/Anti- Symmetric Relation:***
  - A relation $R \subseteq A \times A$ is symmetric if $(b, a) \in R$ whenever $(a, b) \in R$.
  - Let $A = \{1, 2, 3\}$ then
  - $R = \{(1, 2), (2, 1), (2, 3), (3, 2)\}$ is a Symmetric relation defined on set A.
  - ***Note***: if the relation is not symmetric then it is Anti – Symmetric.
  - In mathematics "is equal to" relation between real number is symmetric relation whereas "is less than" is Anti-Symmetric.

Types of Relation (Question)

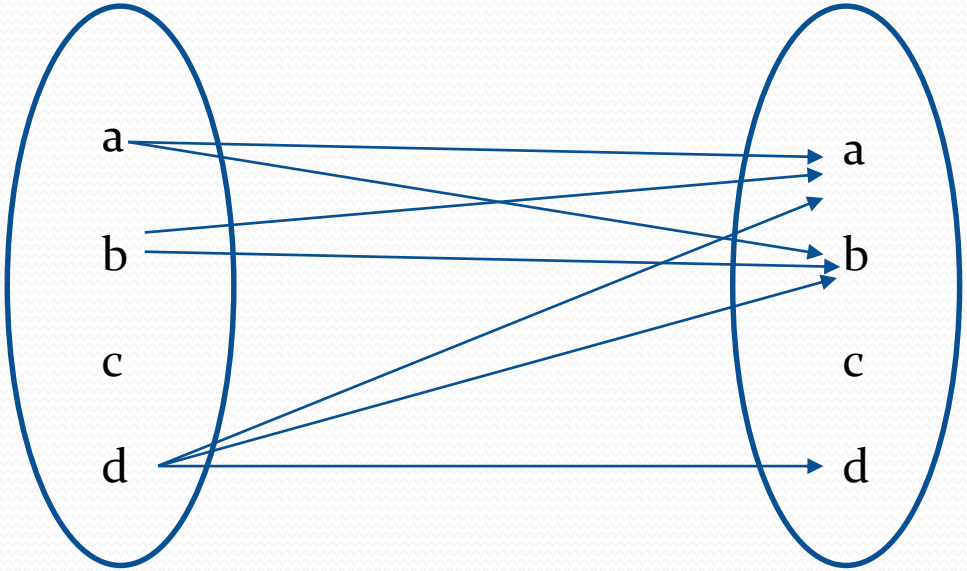
- *Consider a set A={a,b,c,d} and* $R \subseteq A \times A$ such that {(a,a),(b,a),(b,b),(c,c),(d,b)}.
  - Is this relation
    - Reflexive?
    - Symmetric?
- *Consider a set S={1,2,3,4} and T={a,b,c,d} and* $R \subseteq S \times T$ such that {(1,a),(1,c),(2,b),(3,c),(3,d),(4,c),(4,d)}.
  - Is this relation
    - Reflexive?
    - Symmetric?

Types of Relation

- ***Transitive Relation:***
  - A binary relation on set A is transitive relation if a, b, c ∈ A and a is related to b and b is related to c then a is related to C.
  - Let A= { 1, 2, 3} then R= { (1, 2), (2, 3) , (1, 3) } is a transitive relation defined on set A.
  - In mathematics “is greater than” or “is less than” relation between number is transitive relation.

Types of Relation

- ***Consider the relation as follows:***



Is the given relation R

1. ***Reflexive?***
2. ***Symmetric?***
3. ***Transitive?***

Types of Relation

- ***Equivalence Relation:***
  - A relation that is reflexive, symmetric and transitive are called equivalence relation.
- ***Partial Order Relation***:
  - A relation R on a set S is called a “Partial ordering” or a “Partial order”, if R is reflexive, anti-symmetric and transitive.

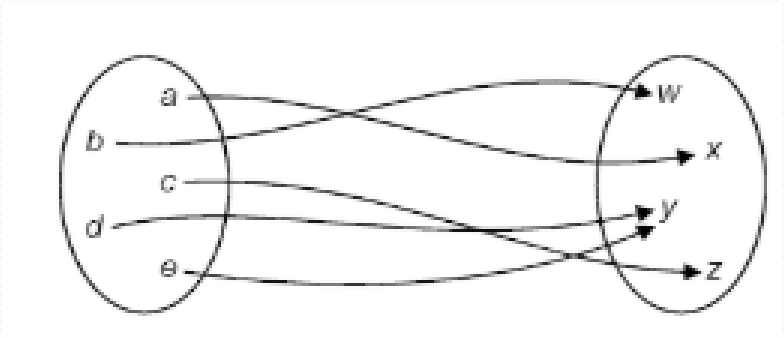
Function

- A function from a set ***A*** to a set ***B*** is a binary relation ***R*** on ***A*** and ***B*** with the following special property:
  - for each element ***a*** $\in$ ***A***, there is exactly one ordered pair in ***R*** with first component ***a***.
- Let ***C*** be the set of cities and ***S*** be the set of states; and let
  - R1 = {(x, y): x $\in$ C, y $\in$ S, and x is a city in state y}
  - R2 = {(x, y): x $\in$ S, y $\in$ C, and y is a city in state x}.

  Here the relation ***R1 is a function***, since each city is in one and only one state, but ***R2 is not a function***, since some states have more than one city.

Function

- Suppose every element of S occurs exactly once as the first element of an ordered pair.
- In Fig shown, every element of S has exactly one arrow arising from it.



S

- This kind of relation is called a “function”.

Function

- A function is said to map an element in its domain to an element in its range.
- Every element in S in the domain, i.e., every element of S is mapped to some element in the range.
- No element in the domain maps to more than one element in the range.
- In general, we use letters such as ***f***, ***g***, and ***h*** for functions and we write $f : A \rightarrow B$ to indicate that ***f*** is a function from A to B.

Function

- If a is any element of A we write $f(a)$ for that element b of B such that $(a, b) \in f$; since f is a function, there is exactly one b $\in$ B with this property, so $f(a)$ denotes a unique object.
- The object $f(a)$ is called the ***image*** of a under f.

Types of Function

- One-to-One Function ***(Injection)***
- Onto Function ***(Surjection)***
- One-to-One Onto function ***(Bijection)***

Types of Function

- One-to-One Function ***(Injection)***
  - A function $f : A \rightarrow B$ is said to be one-to-one if different elements in the domain A have distinct images in the range.

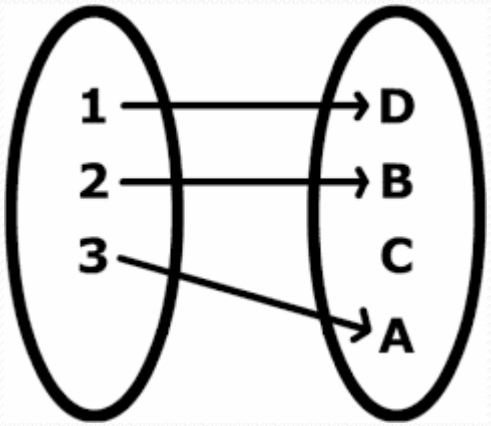


Fig: One to one function

Types of Function

- Onto Function ***(Surjection)***
  - A function $f : A \rightarrow B$ is said to be an onto function if each element of B is the image of some element of A. i.e., $f : A \rightarrow B$ is onto if the image of f is the entire codomain,

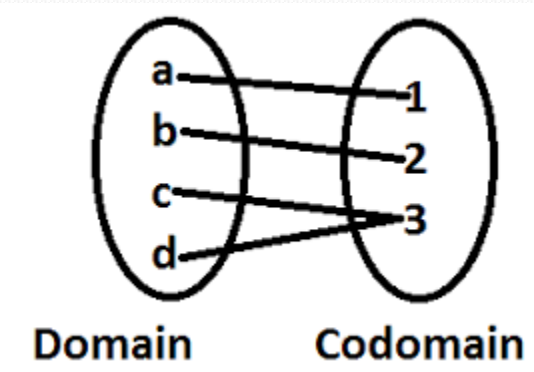


Fig: Onto function

Types of Function

- One-to-One Onto function ***(Bijection)***
  - A function that is both one-to-one and onto is called a "Bijection".
  - Such a function maps each and every element of A to exactly one element of B, with no elements left over.

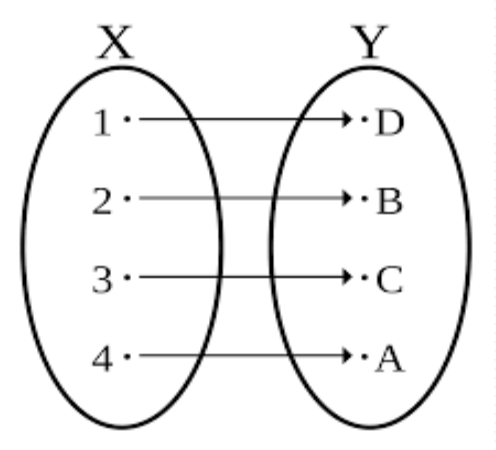


Fig: Bijection function

1.2: Propositional Logic and Proof in Propositional Logic

Logic

- Logic is the generation of idea to solve any problem.
- Logic is study of reasoning and is concerned whether the reasoning is correct or not.
- Logic is a basis or language for reasoning (mathematical and automated).
- Since logic can helps us to reason the mathematical models it needs some rules associated with logic so that we can apply those rules for mathematical reasoning.

Logic

- The rules of logic give precise meaning to mathematical statements and are used to distinguish between valid and invalid mathematical arguments.

Types of Logic

- Propositional Logic
- Predicate Logic
- Fuzzy Logic (Studied in AI……)

Proposition

Proposition

- A declarative statement that is either true or false but not the both are called propositions
- Eg: Today is Friday.

  $2+2=5$

  Pashupati Nath lies in Kaski district.

However following are not propositions

What is your name? *(this is a question)*

Please keep silence. *(this is a request)*

X is even number. *(depends on what X represents)*

Proposition Contd….

Variables are used to represent the propositions.

Eg:

P: Today is Friday.

Q: Kathmandu is capital city of Nepal.

R: It is raining

Here P,Q and R are variables.

Propositional Logic

Propositional Logic

Logic that deals with propositions for reasoning is called propositional logic.

The truth and falsehood of propositions is called its truth value

P: Today is Friday.

P can be either True(T) or False(F) which is the truth values of P.

Truth Table

- A truth table shows how the truth or falsity of a compound statement depends on the truth or falsity of the simple statements from which it's constructed.
- Truth table are sometimes used for proving logical questions.

Types of Propositions

- Simple Propositions
- Compound Propositions

- Simple Propositions are those which contains only one statements.

Compound Propositions

- When multiple propositions are combined together to form a single propositions, then the resulting propositions are called compound propositions.
- To form a compound propositions different connectives are used.
- Thus when a connective is applied to propositions, then resulting proposition is a compound proposition

Connectives

- In propositional logic following connectives are used
  - Negation ($\neg$ or $\sim$)
  - Conjunction ($\wedge$)
  - Disjunction ($\vee$)
  - Implication ($\rightarrow$)
  - Double Implication ($\leftrightarrow$)

Negation

- Let ‘P’ be a proposition then negation of ‘P’ is denoted as ‘¬P’ or ‘∼P’ and read as:

  “not ‘P’.” or “It is not the case that ‘P’.”

P: Today is Friday.

¬P : Today is not Friday.

or

It is not the case that today is Friday.

P	¬P
T	F
F	T

Fig: Truth table of Negation

Conjunction

- Let ‘P’ & ‘Q’ be propositions then conjunction of P,Q is denoted as ‘P∧Q’ and read as:

  “P and Q”

  P: Today is Friday.

  Q: It is raining

  P∧Q: Today is Friday and it is raining.

The truth value of conjunction is true if all the constituent propositions are true otherwise false.

Fig: Truth table of Conjunction

P	Q	P∧Q
T	T	T
T	F	F
F	T	F
F	F	F

Disjunction

- Let ‘P’ & ‘Q’ be propositions then disjunction of P,Q is denoted as ‘P∨Q’ and read as:

  “P or Q”

  P: Today is Friday.

  Q: It is raining

  P∨Q: Today is Friday or it is raining.

  The truth value of disjunction is true if any of the constituent propositions is true otherwise false.

P	Q	P∨Q
T	T	T
T	F	T
F	T	T
F	F	F

Fig: Truth table of Disjunction

Implication

- Let ‘P’ & ‘Q’ be propositions then implication of P and Q is denoted as ‘P→Q’ and read as:

  “ if P then Q ” or “P implies Q”

Hypothesis
(necessary condition)

Conclusion
(sufficient condition)

Some variety of implications

- P *only if* Q
- Q *if* P
- Q *when* P
- Q *is necessary for* p
- Q *provided that* P
- *if* P,Q
- Q *follows from* P
- *a sufficient condition for* Q is P.

All above statements are denoted as P→ Q .

Implication Contd….

Eg: If you get a degree then you can get a job.

P: You get a degree.

Q: You can get a job.

P is ***hypothesis*** or ***sufficient condition***.

Q is ***conclusion*** or ***necessary condition***.

The implication statement will be true if true hypothesis leads to correct conclusion otherwise false .

Truth Table of Implications

Eg.: If you get a degree then you can get a job.

P: You get a degree.

Q: You can get a job

So P→Q will be in truth table as:

P	Q	P→Q
T	T	T
T	F	F
F	T	T
F	F	T

Indirect approach

Fig: Truth table of implication

Double Implication

- Let ‘P’ & ‘Q’ be propositions then implication of P and Q is denoted as ‘P↔Q’ and read as:

  “P if and only if Q”

Here P and Q both are sufficient and necessary conditions

P	Q	P↔Q
T	T	T
T	F	F
F	T	F
F	F	T

Fig: Truth table of double implication

Classwork

- Construct the truth table for the following compound propositions
  - $(P \rightarrow Q) \wedge (Q \rightarrow P)$
  - $(P \rightarrow \neg Q) \leftrightarrow (\neg P \vee Q)$
  - $(\neg P \leftrightarrow R) \vee [(\neg Q \wedge P) \rightarrow \neg R]$

Solutions

$(P \rightarrow Q) \wedge (Q \rightarrow P)$

P	Q	P→Q	Q→P	(P→Q)∧(Q →P)
T	T			
T	F			
F	T			
F	F			

P	Q	P→Q	Q→P	(P→Q)∧(Q →P)
T	T	T	T	T
T	F	F	T	F
F	T	T	F	F
F	F	T	T	T

Solutions

$(P \rightarrow \neg Q) \leftrightarrow (\neg P \vee Q)$

P	Q	¬P	¬Q	P→¬Q	¬P∨Q	(P→¬Q) ↔(¬P∨Q)
T	T					
T	F					
F	T					
F	F					

P	Q	¬P	¬Q	P→¬Q	¬P∨Q	(P→¬Q) ↔(¬P∨Q)
T	T	F	F	F	T	F
T	F	F	T	T	F	F
F	T	T	F	T	T	T
F	F	T	T	T	T	T

Solutions

$(\neg P \leftrightarrow R) \vee [(\neg Q \wedge P) \rightarrow \neg R]$

P	Q	R	$\neg P$	$\neg Q$	$\neg R$	$\neg P \leftrightarrow R$	$\neg Q \wedge P$	$(\neg Q \wedge P) \rightarrow \neg R$	$(\neg P \leftrightarrow R) \vee [(\neg Q \wedge P) \rightarrow \neg R]$
T	T	T							
T	T	F							
T	F	T							
T	F	F							
F	T	T							
F	T	F							
F	F	T							
F	F	F							

Solutions

$(\neg P \leftrightarrow R) \vee [(\neg Q \wedge P) \rightarrow \neg R]$

P	Q	R	$\neg P$	$\neg Q$	$\neg R$	$\neg P \leftrightarrow R$	$\neg Q \wedge P$	$(\neg Q \wedge P) \rightarrow \neg R$	$(\neg P \leftrightarrow R) \vee [(\neg Q \wedge P) \rightarrow \neg R]$
T	T	T	F	F	F	F	F	T	T
T	T	F	F	F	T	T	F	T	T
T	F	T	F	T	F	F	T	F	F
T	F	F	F	T	T	T	T	T	T
F	T	T	T	F	F	T	F	T	T
F	T	F	T	F	T	F	F	T	T
F	F	T	T	T	F	T	F	T	T
F	F	F	T	T	T	F	F	T	T

Converse, Inverse & Contra-Positive

- These terms are defined in terms of conditional statements
- Let $P \rightarrow Q$ be an implication (conditional) statement then:

Inverse :

An inverse of the conditional statement is the negation of both the hypothesis and the conclusion.

The inverse of $P \rightarrow Q$ is $\neg P \rightarrow \neg Q$.

Eg. The inverse of "If she smiles then she is happy" is "If she doesn't smile then she is not happy".

Converse, Inverse & Contra-Positive

Converse :

The converse of the conditional statement is computed by interchanging the hypothesis and the conclusion. If the statement is "If p, then q", the converse will be "If q, then p".

The Converse of $P \rightarrow Q$ is $Q \rightarrow P$.

Eg. The inverse of "If she smiles then she is happy" is "If she is happy then she smiles".

Converse, Inverse & Contra-Positive

Contra-positive *:*

The contra-positive of the conditional is computed by interchanging the hypothesis and the conclusion of the inverse statement. If the statement is "If p, then q", the contra-positive will be "If not q, then not p".

The contra-positive of $P \rightarrow Q$ is $\neg Q \rightarrow \neg P$.

Eg. The contra-positive of "If she smiles then she is happy" is "If she is not happy then she doesn't smile".

Converse, Inverse & Contra-Positive

In Conclusion:

- Let $P \rightarrow Q$ be an implication statement then:
- A statement $Q \rightarrow P$ is called its converse.
- A statement $\neg P \rightarrow \neg Q$ is called its inverse.
- A statement $\neg Q \rightarrow \neg P$ is called its contra-positive.

“If you do your homework, you will not be punished.” *(Implication)*
"If you will not be punished, you do your homework”. *(converse)*
“If you do not do your homework, you will be punished.” *(inverse)*
"If you are punished, you did not do your homework”. *(contra-positive)*

Types of Compound Statements

- Compound statements can be classified according to truth table as:
  - Tautology
  - Contradiction
  - Contingency

Tautology

- A compound statement is said to be tautology if its truth values are always true no matter what the truth values of their constituent propositions.
- Irrespective to the truth values of constituent propositions, the truth values of compound propositions are always true.
- Eg. $P \vee \neg P$ is a tautology

P	¬P	P∨¬P
T	F	T
F	T	T

Example Contd…..

- Show that following compound propositions are tautology:
  - $P \rightarrow (P \vee Q)$
  - $[P \wedge (P \rightarrow Q)] \rightarrow Q$

Solutions

- $P \rightarrow (P \vee Q)$

P	Q	P∨Q	P→(P∨Q)
T	T		
T	F		
F	T		
F	F		

P	Q	P∨Q	P→(P∨Q)
T	T	T	T
T	F	T	T
F	T	T	T
F	F	F	T

Solutions

- $[P \wedge (P \rightarrow Q)] \rightarrow Q$

P	Q	$P \rightarrow Q$	$P \wedge (P \rightarrow Q)$	$[P \wedge (P \rightarrow Q)] \rightarrow Q$
T	T			
T	F			
F	T			
F	F			

P	Q	$P \rightarrow Q$	$P \wedge (P \rightarrow Q)$	$[P \wedge (P \rightarrow Q)] \rightarrow Q$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

Home Work

- $[(P \rightarrow Q) \wedge (Q \rightarrow R)] \rightarrow (P \rightarrow R)$ show this compound proposition is a tautology using truth table.

Contradiction

- A compound proposition whose truth values are always false are called contradiction.
- Eg. $P \wedge \neg P$ is a contradiction

P	¬P	P∧¬P
T	F	F
F	T	F

Contingency

- Contingency is a compound proposition whose truth values are combination of both true and false.
- Eg. $\neg P \vee Q$ is a contingency

P	Q	¬P	¬P∨Q
T	T	F	T
T	F	F	F
F	T	T	T
F	F	T	T

Logical Equivalence

- Let P and Q be two compound propositions then P and Q are said to be logically equivalent if their truth values are always same under same condition of truth values of constituent propositons.
- Eg.

Implication and its contra-positive are logically equivalent.

i.e: $P \rightarrow Q \equiv \neg Q \rightarrow \neg P$

Logical Equivalence

$P \rightarrow Q \equiv \neg Q \rightarrow \neg P$

P	Q	¬P	¬Q	P→Q	¬Q→¬P
T	T	F	F	T	T
T	F	F	T	F	F
F	T	T	F	T	T
F	F	T	T	T	T

Question to Practice

- Show that following compound propositions are logically equivalent using truth table.
  - $P \leftrightarrow Q \equiv [(P \rightarrow Q) \wedge (Q \rightarrow P)]$
  - $[P \vee (Q \wedge R)] \equiv [(P \vee Q) \wedge (P \vee R)]$

Translating Sentences into Statements of Propositional Logic

- There are simple three steps to translate the sentences into propositional logic statements
  - Identify all the individual sentences in the given sentence and represent them by different variables.
  - Identify all the connectives used in given sentence.
  - Write an expression in terms of variables and connectives.

  Note: we assume there is no negative statements in the given sentences
- Obtained compound propositional will be the required statement of propositional logic.

Translating Sentences into Statements of Propositional Logic

- *Eg:. You can access the college internet only if you are a computer science student or you are not a fresher.*

Solution

- Individual sentences are:
  - P: You can access the college internet
  - Q: You are a computer science student
  - R: You are a fresher
- Connectives are:
  - You can access the college internet *only if* you are a computer science student *or* you are *not* a fresher.
    - Only if i.e implication (→)
    - Or i.e disjunction (∨)
    - Not i.e negation(¬)
- Propositional Logic representation:

$$P \rightarrow (Q \vee \neg R)$$

Translating Sentences into Statements of Propositional Logic

Hiking is safe along the trail if and only if berries are ripe along the trail and bears have not been seen along the trail.

Solution

- Individual sentences are:
  - P: Hiking is safe along the trail
  - Q: Berries are ripe along the trail
  - R: Bears have been seen along the trail.
- Connectives are:

  Hiking is safe along the trail *if and only if* berries are ripe along the trail *and* bears have *not* been seen along the trail.
  - If and only if i.e double-implication ($\leftrightarrow$)
  - and i.e conjunction ($\wedge$)
  - Not i.e negation($\neg$)
- Propositional Logic representation:

$$P \leftrightarrow (Q \wedge \neg R)$$

Rules of Inference for Propositional Logic

- Arguments:
  - An argument in propositional logic is the sequence of propositions.
  - The last statement is the *conclusion* and all its preceding statements are called *premises* (or hypothesis).
  - *The argument is valid if the premises imply the conclusion*
- A valid argument is one where the conclusion follows from the truth values of the premises.

Rules of Inference for Propositional Logic

- To deduce new statements from the statements whose truth that we already known, ***Rules of Inference*** are used.
- Rules of Inference in propositional logic are:
  - Addition
  - Simplification
  - Modus Ponens
  - Modus Tollens
  - Hypothetical Syllogism
  - Disjunctive Syllogism
  - Conjunction
  - Resolution

Rules of Inference for Propositional Logic

Addition:

- *If P is a premise, we can use Addition rule to derive $P \vee Q$*

$$\frac{P}{\therefore P \vee Q}$$

Example: P: "I will study TOC."
Q: "I will study Maths."

"I will study TOC" is true

So, I will study TOC or I will Study Maths." is true.

Rules of Inference for Propositional Logic

Simplification:

- If P∧Q is a premise, we can use simplification rule to derive P.

$$\frac{P \wedge Q}{\therefore P}$$

Example:

"He studies very hard and he gets best grade": $P \wedge Q$

P: He studies very hard.

Q: He gets best grade

So, "He studies very hard." P is true.

"He gets best grade." Q is true.

Rules of Inference for Propositional Logic

Modus Ponens:

- If P→Q and P are two premises, we can use Modus Ponens to derive Q.

$$\frac{P \rightarrow Q \qquad P}{\therefore Q}$$

Example:"If it snows today, then we will go skiing".
$P \rightarrow Q$

The hypothesis " it snows today," P

Then by modus ponens, "We will go skiing." Q is true

Rules of Inference for Propositional Logic

Modus Tollens:

- If P→Q and ¬Q are two premises, we can use Modus Tollens to derive ¬P.

$$\frac{P \rightarrow Q \quad \neg Q}{\therefore \neg P}$$

"If you have a password, then you can log on to facebook." $P \rightarrow Q$

"You cannot log on to facebook." $\neg Q$

Therefore – "You do not have a password ." $\neg P$

Rules of Inference for Propositional Logic

Disjunctive Syllogism:

- If ¬P and P∨Q are two premises, we can use Disjunctive Syllogism to derive Q.

$$\frac{\neg P \qquad P \vee Q}{\therefore Q}$$

"The ice cream is not vanilla flavored". $\neg P$

"The ice cream is vanilla flavored or chocolate flavored". $P \vee Q$

Therefore "The ice cream is chocolate flavored". Q

Rules of Inference for Propositional Logic

Hypothetical Syllogism :

- If P→Q and Q→R are two premises, we can use Hypothetical Syllogism to derive P→R

$$\frac{\begin{array}{c} P \rightarrow Q \\ Q \rightarrow R \end{array}}{\therefore P \rightarrow R}$$

Example:

"If it rains, I will not go to school", *P→Q*

"If I don't go to school, I won't need to do homework". *Q→R*

Therefore – "If it rains, I won't need to do homework." *P→R*

Rules of Inference for Propositional Logic

Conjunction :

- If P and Q are two premises, we can use conjunction to derive P∧Q

$$\frac{P \quad Q}{\therefore P \wedge Q}$$

Example:

"He is good at study", *P*

"He gets highest marks in TOC". *Q*

Therefore – "He is good at study and he gets highest marks in TOC." *P∧Q*

Rules of Inference for Propositional Logic

Resolution :

- If $P \vee Q$ and $\neg Q \vee R$ are two premises, we can use resolution to derive $P \vee R$

$$\begin{array}{l} P \vee Q \\ \neg Q \vee R \\ \hline \therefore \quad P \vee R \end{array}$$

"I will study TOC or I will study Maths ." $P \vee Q$
"I will not study Math or I will study Data Science." $\neg Q \vee R$
Therefore, "I will study TOC or I will study Data Science."
$P \vee R$

In Conclusion, rules of Inference are tabulated as:

Rules	Name	Rules	Name
$\dfrac{P}{\therefore P \vee Q}$	Addition	$\dfrac{P \quad Q}{\therefore P \wedge Q}$	Conjunction
$\dfrac{P \wedge Q}{\therefore P}$	Simplification	$\dfrac{\neg P \quad P \vee Q}{\therefore Q}$	Disjunctive Syllogism
$\dfrac{P \to Q \quad P}{\therefore Q}$	Modus Ponens	$\dfrac{P \to Q \quad Q \to R}{\therefore P \to R}$	Hypothetical Syllogism
$\dfrac{P \to Q \quad \neg Q}{\therefore \neg P}$	Modus Tollens	$\dfrac{P \vee Q \quad \neg Q \vee R}{\therefore \quad P \vee R}$	Resolution

Question No. 1

- Using rules of inference show the following hypothesis :
  - *It is not sunny this afternoon and it is colder than yesterday.*
  - *We will go swimming only if it is sunny.*
  - *If we don't go swimming , then we will go for a canoe trip.*
  - *If we go for a canoe trip, then we will be home by sunset.*

    Leads to conclusion:
  - *We will be home by sunset.*

Solution

- Solution Steps:
- *Convert the given sentences into propositional logic statement.*
- *Identify the hypothesis and conclusion.*
- *Use rules of inference in hypothesis to prove the conclusion.*

Solution Contd….

Identifying individual statements

- *p : It is sunny this afternoon.*
- *q : It is colder than yesterday.*
- *r : We will go swimming .*
- *s : we will go for a canoe trip.*
- *t : We will be home by sunset.*

- Identifying and writing hypothesis and conclusion in propositional logic statements

- Hypothesis are:
  - $\neg p \wedge q$
  - $r \rightarrow p$
  - $\neg r \rightarrow s$
  - $s \rightarrow t$
- Conclusion:
  - t

Hypothesis:

- *It is not sunny this afternoon and it is colder than yesterday.*
- *We will go swimming only if it is sunny.*
- *If we don't go swimming , then we will go for a canoe trip.*
- *If we go for a canoe trip, then we will be home by sunset.*

Conclusion:

We will be home by sunset.

Hypothesis are: $\neg\ p \wedge q$
$r \rightarrow p$
$\neg r \rightarrow s$
$s \rightarrow t$

Conclusion: t

- Proof:

S.No.	Steps	Reasons
1	$\neg\ p \wedge q$	Given hypothesis
2	$\neg\ p$	Using simplification on 1.
3	$r \rightarrow p$	Given hypothesis
4	$\neg r$	Using modus tollens on 2 and 3.
5	$\neg r \rightarrow s$	Given hypothesis
6	s	Using modus ponens on 4 and 5.
7	$s \rightarrow t$	Given hypothesis
8	t	Using modus ponens on 6 and 7

Hence the given hypothesis leads to the conclusion.

Question No. 2

- Using rules of inference show the following hypothesis :
  - *If you send me an email message then I will finish writing the program.*
  - *If you don't send me an email message then I will go to sleep early*
  - *If I go to sleep early then I will wake up feeling refreshed.*

    Leads to conclusion:
  - *If I don't finish writing the program then I will wake up feeling refreshed.*

Solution

- Solution Steps:
- *Convert the given sentences into propositional logic statement.*
- *Identify the hypothesis and conclusion.*
- *Use rules of inference in hypothesis to prove the conclusion.*

Solution Contd….

Identifying individual statements

- *p : you send me an email message .*
- *q : I will finish writing the program.*
- *r : I will go to sleep early*
- *s : I will wake up feeling refreshed.*

- Identifying and writing hypothesis and conclusion in propositional logic statements

- Hypothesis are:
  - $p \rightarrow q$
  - $\neg p \rightarrow r$
  - $r \rightarrow s$
- Conclusion:
  - $\neg q \rightarrow s$

Hypothesis:

❖ *If you send me an email message then I will finish writing the program.*

❖*If you don't send me an email message then I will go to sleep early*

❖*If I go to sleep early then I will wake up feeling refreshed.*

Conclusion:

If I don't finish writing the program them I will wake up feeling refreshed.

Solution Contd….

Hypothesis are: $p \rightarrow q$

$\neg p \rightarrow r$

$r \rightarrow s$

Conclusion: $\neg q \rightarrow s$

- Proof:

S.No.	Steps	Reasons
1	$p \rightarrow q$	Given hypothesis
2	$\neg q \rightarrow \neg p$	Using contra-positive on 1.
3	$\neg p \rightarrow r$	Given hypothesis
4	$\neg q \rightarrow r$	Using hypothetical syllogism on 2 and 3.
5	$r \rightarrow s$	Given Hypothesis
6	$\neg q \rightarrow s$	Using hypothetical syllogism on 4 and 5.

Hence the given hypothesis leads to the conclusion.

Question No. 3

Using rules of inference show the following hypothesis:

- If Clinton does not live in France, then he does not speak French.
- Clinton does not drive a Datsun.
- If Clinton lives in France, then he rides a motorcycle.
- Either Clinton speaks French or he drives a Datsun.

Leads to the conclusion

- Clinton ride a motorcycle.

Question No. 4

Show that the premises “If my cheque book is in office, then I have paid my phone bill”, “I was looking for phone bill at breakfast or I was looking for phone bill in my office”, “If was looking for phone bill at breakfast then my cheque book is on breakfast table” , “If I was looking for phone bill in my office then my cheque book is in my office”, “I have not paid my phone bill” imply the conclusion “My cheque book is on my breakfast table”

Limitation of propositional Logic

Consider the following two statements:

- Every BCT student must study TOC.
- Pratik is BCT student.

- It looks “logical” to derive:
  - Pratik must study TOC.
- However, this cannot be expressed by propositional logic because we already notice that none of the logical operators we have learnt are applicable here.
- i.e Propositional Logic is not expressive.

Limitation of propositional Logic

Consider the statements containing variables as:

a) $x>5$
b) $x=y+2$

- These statements are neither true nor false when the value of variables are not specified.
- The statement x is greater than 5 has two parts:
  - *the variable x which is subject of statement*.
  - *the second part "is greater than" is a predicate which is the property of subject of statement.*
- To express such statements we need powerful logical tool called *predicate logic.*

Predicate Logic

- It is also known as First Order Propositional Logic (FOPL).
- Predicate logic is an extension of Propositional logic.
- It adds the concept of predicates and quantifiers to better capture the meaning of statements that cannot be expressed by propositional logic.

Predicate Logic Contd….

Moving back to statement: x>5.

- We can denote x is greater than 5 as:

  P(x), where *P* is a predicate which denotes *"is greater than 5"* and *x is a variable*

- The statement P(x) is also said to be value of propositional function *P* at *x*.
- When the value of variable i.e. *x* is assigned the statement *P(x)* becomes proposition.
- Here value of *x* is assigned from domain. Eg: natural numbers.

Predicate Logic Contd….

Let P(X) denotes “X>5” what is the truth values of P(2) and P(7) ?

statement P(2) is obtained by assigning X=2 in “X>5”.

hence “2>5” is false

Similarly

statement P(7) is obtained by assigning X=7 in “X>5”.

hence “7>5” is true

Predicate Logic Contd….

For statement x=y+2

predicate is defined as P(x,y):"x=y+2"

So,

P(5,3) is true.

P(6,7) is false.

Here *x* and *y* are taken from some specific domain.

Quantifier are used to convert the predicate logic into propositional logic.

Quantification

- The process of binding the propositional variable over given domain is called quantification.
- Quantification is needed to identify the truth value of the predicate.
- Two types of Quantification:
  - Universal Quantification
  - Existential quantification

Universal Quantification

- Let P(X) be a predicate then universal quantification of P(X) is denoted by:

  $\forall x\ P(X)$

  and read as “for all X P(X)” or “for every X P(X)”

  Here ∀ is called universal quantifier.

  The truth values of $\forall x\ P(X)$ is true if P(X) is true for all the values of the given domain and is false if P(X) is false for at least one value of the given domain.

Universal Quantification Contd….

Let P(X1,X2,……..Xn) be a predicate such that P(X1), P(X2)….P(Xn) are different instances.

Then ∀x P(X1,X2,……..Xn) is true if all the above instances are true and is false if any of the above instance is false

∀x P(X1,X2,……..Xn)≡ P(X1) ∧ P(X2) ∧ ……∧P(Xn)

Universal Quantification Contd….

Eg: Let $P(X)=X^2-1>0$, $X \in R$ What is the truth values of $\forall x P(X)$?

Solution,

When $X=1$, $P(1)=1^2-1>0$

$=0>0$ which is false

So by definition of universal quantification the truth value of $\forall x P(X)$ is false.

The value of the variable that makes the universally quantified statement false are called counter example. Here is this example X=1 is counter example.

Existential Quantification

- Let P(X) be a predicate then existential quantification of P(X) is denoted by:

  $\exists x\ P(X)$

  and read as “there exist X P(X)” or “for some X P(X)”

  Here ∃ is called existential quantifier.

  The truth values of $\exists x\ P(X)$ is true if P(X) is true for at least one the values of the given domain and is false if P(X) is false for each value of the given domain.

Existential Quantification Contd….

Let P(X1,X2,……..Xn) be a predicate such that P(X1), P(X2)….P(Xn) are different instances.

Then ∃x P(X1,X2,……..Xn) is true if all the above instances are true and is false if any of the above instance is false

∃ x P(X1,X2,……..Xn)≡ P(X1) ∨P(X2) ∨ …… ∨ P(Xn)

Existential Quantification Contd….

Eg: Let $P(X)=X^2-1=0$, $X \in R$ What is the truth values of $\exists x P(X)$?

Solution,

When $X=1$, $P(1)=1^2-1=0$

$=0=0$ which is true

So by definition of existential quantification the truth value of $\exists x P(X)$ is true.

The value of the variable that makes the existentially quantified statement true are called witness. Here is this example X=1 is witness.

1.3: Proof

Proof Techniques

- Induction Principle
- Pigeonhole Principle
- Diagonalization Principle

Induction Principle

- Mathematical induction is an important proof technique that can be used to prove mathematical theorems or statements
- Let p(n) be a statement. Now our concern is to show that p(n) is true using mathematical induction.
- For this we first show that p(n) is true for some initial value like n= 0,1,2,… ***This step is called basic step.***
- Then we assume that p(n) is true for any arbitrary value 'k' i.e. p(k) is true and show that p(n) is true for 'k+1" i.e. p(k+1) true. ***This step is called inductive step.***

Induction Principle

- The principle of mathematical induction is applicable for the statement containing natural numbers including zero.

Induction Principle (Example_1)

- Using mathematical induction show that

$$1+2+3+4+\ldots\ldots\ldots\ldots\ldots = \frac{n(n+1)}{2}$$

Solution,

$$P(n)=1+2+3+4+\ldots\ldots\ldots\ldots\ldots = \frac{n(n+1)}{2} \text{ is true.}$$

Basic Step:

$$P(n)=1+2+3+4+\ldots\ldots\ldots\ldots\ldots = \frac{n(n+1)}{2}$$

for n=1, $P(1)= \frac{1(1+1)}{2}=1$, which is true.

for n=2, $P(2)= \frac{2(2+1)}{2}=3$, which is also true.

Induction Principle (Example_1)

Inductive Step:

We assume that P(n) is true for any arbitrary value 'k' i.e. P(k) is true and show that P(n) is true for 'k+1' i.e. P(k+1) is true.

let, $P(k)=\frac{k(k+1)}{2}$ is true.

Now we try to prove for k+1

$$P(k+1) = 1+2+3+4+\ldots\ldots+k+(k+1) = \frac{(k+1)((k+1)+1)}{2}$$

$$= \frac{k(k+1)}{2}+(k+1)$$

$$= \frac{k(k+1)+2(k+1)}{2}$$

Induction Principle (Example_1)

$$= \frac{(k+1)(k+2)}{2}$$

$$= \frac{(k+1)(k+1+1)}{2}$$

$$= \frac{(k+1)((k+1)+1)}{2}$$

This shows that when P(k) is true, P(k+1) is also true.

Thus, by the principle of mathematical induction, P(n) is true for all n.

Induction Principle (Example_2)

- Use the principle of Mathematical Induction to verify that , for n any positive integer, n^4-4n^2 is divisible by 3 for n >=0.

Solution,

$P(n)= n^4-4n^2$ is divisible by 3 for n>=0.

Basic Step:

$P(n)= n^4-4n^2$

for n=0, $P(0)=0^4-4(0)^2=0$, which is divisible by three is true.

for n=1, $P(1)=1^4-4(1)^2=-3$, which is divisible by three is true.

Induction Principle (Example_2)

Inductive Step:

We assume that P(n) is true for any arbitrary value 'k' i.e. P(k) is true and show that P(n) is true for 'k+1' i.e. P(k+1) is true.

let, $P(k)= k^4-4k^2$ is true.

Now we try to prove for k+1

$(k+1)^4 -4(k+1)^2$ is divisible by 3.

$= k^4+4k^3+6k^2+4k+1-\{4(k^2+2k+1)\}$

$= (k^4-4k^2)+4k^3+6k^2+4k+1-8k-4$

$= (k^4-4k^2)+4k^3+6k^2-4k-3$

$= (k^4-4k^2)+4(k^3-k)+3(k^2-1)$

Induction Principle (Example_2)

$= (k^4-4k^2)+4(k^3-k)+3(k^2-1)$

Here, **(k^4-4k^2)** is divisible by 3 is assumed to be true.

Again, (k^3-k) is divisible by 3 i.e. (k-1)*k*(k+1), which is the product of three consecutive natural number so it must be divisible by 3. Hence **$4(k^3-k)$** is divisible by 3.

Further, **$3(k^2-1)$** is divisible by 3.

Hence we can conclude that the sum of the term which are individually divisible by 3 is obvious divisible by 3. i.e. $(k^4-4k^2)+4(k^3-k)+3(k^2-1)$ is divisible by 3.

This shows that when P(k) is true, P(k+1) is also true.

Thus, by the principle of mathematical induction, P(n) is true for all n>=0.

Induction Principle (Example_3)

- Prove by mathematical induction that $n < 2^n$ for all positive integers n.

Solution,

$P(n)= n < 2^n$ is valid for all positive integers n.

Basic Step:

$$P(n)= n < 2^n$$

for n=1, $P(1)= 1 < 2^1 = 1 < 2$ which is true.

Induction Principle (Example_3)

Inductive Step:

we assume that P(n) is true for any arbitrary value 'k' i.e. P(k) is true and show that P(n) is true for 'k+1" i.e. P(k+1) is true.

let, $P(k)=k < 2^k$ is true.

Now we try to prove for k+1

i.e. $P(k+1)=k+1 < 2^{k+1}$ is true.

We have, $k < 2^k$

Multiplying both side by 2,

or, $2*k < 2*2^k$

or, $2*k < 2^{k+1}$

Induction Principle (Example_3)

or, $k+k < 2^{k+1}$ ..... (1)

Here k is the positive integer and for n=1, it is shown true already.

So, we need to prove it for

$1 < k$

Adding k on both side,

$1+k < k+k$ ..... (2)

Comparing (1) and (2),

$1+k < 2^{k+1}$

This shows that when P(k) is true, P(k+1) is also true.

Thus, by the principle of mathematical induction, P(n) is true for all n.

Induction Principle (CW)

- Using mathematical induction prove that the statement $6*7^n-2*3^n$ is divisible by 4 for n=1,2,3,.....
- Prove the following statement by using mathematical induction:

  $1*1!+2*2!+3*3!+\ldots\ldots\ldots+n*n! = (n+1)!-1$ where $(n>=1)$
- Use principle of mathematical induction to prove (5^n-1) is divisible by 4 for all integers n > 0.
- Use mathematical induction to show that $2^n < n!$ for any positive integer n >= 4.

Pigeonhole Principle

- ***If A and B are finite sets and $|A| > |B|$, then there is no one-to-one function from A to B.***
- That is, if an attempt is made to pair of the elements of A (the "pigeons") with elements of B (the "pigeonholes"), sooner or later we will have to put more than one pigeon in a pigeonhole.
- The pigeonhole principle states that if n pigeons are put into m pigeonholes with n>m, then at least one pigeonhole must contain more than one pigeon.

Pigeonhole Principle

- Pigeonhole principle can be used to show that certain languages are not regular.
- We will use pigeonhole principle in the topic pumping lemma later.

Diagonalization Principle

- *Let R be a binary relation on a set A, and the diagonal set for R denoted by D, be {a:a ∈ A and (a, a) ∉ R}. For each a ∈ A, let* R_a *= {b: b ∈ A and (a, b) ∈ R}. Then D is distinct from each* R_a.
- If ***A*** is a finite set, then ***R*** can be pictured as a square array; the rows and columns are labeled with the elements of ***A*** and there is a cross in the box with row labeled ***a*** and column labeled ***b*** just in case ***(a, b) ∈ R***.
- The diagonal set ***D*** corresponds to the complement of the sequence of boxes along the main diagonal, boxes with crosses being replaced by boxes without crosses, and vice versa.

Diagonalization Principle

- The sets $\boldsymbol{R_a}$ correspond to the rows of the array.
- The diagonalization principle can then be rephrased: ***"The complement of the diagonal is different from each row."***
- Let us consider the set $\boldsymbol{A=\{a,b,c,d,e,f\}}$ and relation $\boldsymbol{R}$ *{(a, b), (a, d), (b, b), (b, c), (c, c), (d, b), (d, c), (d, e), (d, f), (e, e), (e, f), (f, a), (f, c), (f, d), (f, e) }; notice that* $R_a = \{b,d\}$, $R_b = \{b,c\}$, $R_c = \{c\}$, $R_d = \{b,c,e,f\}$, $R_e = \{e,f\}$ *and* R_f *{a.c.d,e}.*

Diagonalization Principle

- All in all, ***R*** may be pictured like this:

	a	*b*	*c*	*d*	*e*	*f*
a		×		×		
b		×	×			
c			×			
d		×	×		×	×
e					×	×
f	×		×	×	×	

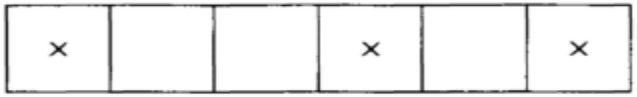
The sequence of boxes along the diagonal is

	×	×		×	

Its complement is

×			×		×

Diagonalization Principle



- This corresponds to the diagonal set D = { a, d, f}.
- Indeed, D is different from each row of the array; for D, because of the way it is constructed, differs from the first row in the first position, from the second row in the second position, and so on.
- The diagonalization principle holds for infinite sets as well, for the same reason: The diagonal set D always differs from the set R_a on the question of whether a is an element, and hence cannot be the same as R_a for any a.

Diagonalization Principle

- The diagonalization principle to make the proof are as follows:
  - Assume condition for contradiction.
  - Find the reversed diagonal and check whether it is different from each row in table or not.
  - If it is different from each row in table, it contradict the assumed condition proving the theorem.
- ***The set of real numbers in {0, 1} is uncountable can be proved using diagonalization principle.***

Diagonalization Principle

- In the theory of computation, diagonalization is a fundamental technique used to show that certain problems are undecidable or uncomputable.
- The basic idea behind diagonalization is to construct a problem that cannot be solved by any algorithm or Turing machine.
- One famous example of diagonalization is the proof that the halting problem is undecidable.
- The halting problem is the problem of determining, given a program and an input, whether the program will eventually halt or run forever.

Diagonalization Principle

- Alan Turing proved that there is no algorithm that can solve the halting problem for all possible programs and inputs.
- Another example of diagonalization is the proof that there are infinitely many prime numbers.
  - This proof uses a technique called Euclid's proof by contradiction, which involves assuming that there are only a finite number of primes and then constructing a new prime that contradicts that assumption.
- Overall, diagonalization is a powerful tool in the theory of computation that allows us to prove important results about the limits of computation.

1.4: Alphabets, Language and Operations

Alphabet

- Alphabet is a finite non-empty set of symbols.
- The symbols can be the letters such as {a, b, c}, bits {1, 0} ,digits {0,1,2,3……..9}, common characters like $, #, etc.
- It is denoted by Σ.
  - *Σ={0,1} is a binary alphabet.*
  - *Σ={0,1,2,….9} is a decimal alphabet.*
  - *Σ={a, b, c,……, z, A, B, C,……..Z} is an English alphabet.*

String

- String is a finite sequence of symbols taken from some alphabet (Σ), where each symbol is element of Σ.
- It is denoted by w.
  - Example: Strings over the binary alphabet Σ{ 0, 1 } can be w={0,1,00,11,001,1101,1111}
  - w={aab, abcb, b, cc} are four strings over the alphabet Σ { a, b, c }.

String

- The number of symbols in a string w is called its length, denoted by |w|.
- If W=aabab then |w| is 5.
- The empty string denoted by ε(epsilon) is string having length zero. i.e w= ε then |w| =0.
- The set of all string over an alphabet is denoted by Σ^*.
  - Example ∑={a,b,c} then

$$\sum^* = \{\varepsilon, a, b, c, aa, ab, bc, ca, aca, aacb, \ldots\ldots\}$$

Power of Alphabet

- The set of all strings of certain length k from an alphabet is the k^{th} power of alphabet i.e.:

$$\sum^k = \{w: |w| = k\}$$

- For $\sum=\{0,1\}$
  - $\sum^0 = \{\varepsilon\}$
  - $\sum^1 = \{0,1\}$
  - $\sum^2 = \{00, 01, 10, 11\}$
  - $\sum^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

Kleen Closure

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

$$\sum^* = \sum^0 \cup \sum^1 \cup \sum^2 \cup \sum^3 \ldots\ldots\ldots$$

- If ∑={a} then ∑*= {a^n, n=0,1,2,3.....}

∑*= {ε, a, aa, aaa, aaaa, ...}

Positive Closure

- The set of all the strings over an alphabet $\sum$ is called positive closure of $\sum$ except ε(epsilon) and is denoted by $\sum^+$,.

$$\sum^+ = \sum^1 \cup \sum^2 \cup \sum^3 \ldots\ldots\ldots$$

- If $\sum$={a} then $\sum^+$= {a^n, n=1,2,3.....}

$\sum^+$= {a, aa, aaa, aaaa, ...}

String Operation

- ***String Reversal:***
  - For any string w, its reversal denoted. w^R, is a string spelled backward.
  - Example: w={bbabba} then w^R ={abbabb}.
- ***String Concatenation:***
  - Two strings over the same alphabet can be combined to form a third by operation of concatenation.
  - The concatenation of strings ***x*** and ***y*** written ***xoy*** or simply ***xy*** is the string ***x*** followed by the string ***y***.
  - *If x=aba and y=bba then xy=ababba and yx=bbaaba*
  - ***Note: concatenating the empty string ε with another string, the result is just the other string.***

String Operation

- ***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: If w= abcd then,

    s=bcd, s=cd, s=d are suffix of w
  - s is proper suffix if $s \neq w$.
- ***Prefix of a string:***
  - String is called prefix of a string w if it is obtained by removing zero or more trailing symbols in w.
  - For example: If w= abcd then,

    s=a, s=ab, s=abc are prefix of w
  - s is proper prefix if $s \neq w$.

String Operation

- ***Substring:***
  - A strings ***s*** is called substring of a string ***w*** if it is obtained by removing one or more leading or trailing symbols in w.
  - It is proper substring if ***s ≠ w.***
  - A string ***v*** is substring of a string ***w*** if and only if there are strings ***x*** and ***y*** such that w=***xvy***
  - In other word ***s*** is a substring of ***w*** if ***s*** appears consecutively within ***w***. e.g " ***deck***" is a substring of abc***deck***abcjkl.

Language

- A language denoted by ***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} that ends with 1.
L={1, 01 ,11, 011, 0111,.....}

→ English language is a set of strings taken from the alphabet ={a, b, c,.....,z, A, B, C,.....Z}.
L= {dbms, TOC, Graphics}.

Language

- Example

  For $\sum = \{ 0, 1\}$

  $L = \{ x : x \in \{ 0,1 \}^* \mid x$ has even number of Zero's $\}$

  then $01101100 \in L$, $1111 \in L$, $1011011000 \notin L$.

- The infinite language L are denoted as

  $L = \{ w \in \sum^* : w$ has property P$\}$

Empty Language

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

Membership in Language

- Membership in a language defines association of some string within 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 on L and for w= "HELLO" the membership of w is false in L.

Regular Expression

- Regular Expression is a formula for representing a language in terms of a form combined using 3 operations union, concatenation and kleen closure.
- A regular expression is a string that describe a whole set of strings, according to certain syntax rules.
- Any regular expression is recursively defined as:
  - Empty state (ϕ), empty string(ε) symbol of input alphabet ($\sum$) are regular expression
  - Let R1 and R_2 be regular expression then union of R1 and R2 denoted by (R1 + R2) is also Regular Expression.

Regular Expression

- Let R1 and R_2 be regular expression then concatenation of R1 and R2 denoted by (R1 . R2) is also Regular Expression.
- Let R be regular expression then kleen closure of R denoted by R* is also Regular Expression.

- ***Note: Every regular expression is associated with some language.***

Regular Expression

- Example of regular expression, Assume that ∑ = {a, b, c}
  - Zero or more: ***a**** means "zero or more a's",
  - To say "zero or more ab's," i.e., {ε, ab, abab, ...} you need to say ***(ab)****.
  - One or more: Since a* means "zero or more a's", you can use ***aa**** (or equivalently ***a*a***) to mean "one or more a's".
  - Similarly to describe 'one or more ab's", that is {ab, abab, ababab, ...}, you can use ***ab(ab)****.
  - Zero or one: It can be described as an optional 'a' with ***(a + ε)***.
  - Any string at all: To describe any string at all (with ∑ = { a, b, c} you can use ***(a + b + c)****.

Regular Expression

- Any non-empty string: This is written any character from ∑ = {a, b, c} followed by any string at all: $(a + b + c)(a + b + c)^*$
- Any string not containing ……: To describe any string at all that does not contain an 'a' (with ∑ ={a, b, c}), you can use $(b + c)^*$.
- Any string containing exactly one ……: To describe any string that contains exactly one 'a' (with ∑ ={a, b, c}) put "any string not containing an a", on either side of the 'a' like: $(b + c)^* a (b + c)^*$ **or** $a(b+c)^*$ **or** $(b+c)^*a$

Regular Expression

- **Example:**

$(0+1)^*=$

$0^*+1^*=$

$(01)^*=$

$0^* \cdot 1^*=$

$1.1^*=$

Regular Expression

$(0+1)^*=\{ \in,0,1,00,101,1101,00101,.....\}$

$0^*+1^*=\{ \in,0,1,00,11,111,0000,.....\}$

$(01)^*=\{\in,01,0101,010101,.....\}$

$0^*.1^*= \{ \in,0,1,00,11,000,01,001,0111.....\}$

$1.1^*=\{1,11,111,11111,.....\}$

Regular Expression

- Precedence of Regular-Expression Operators
  - The star Operator (*) is of highest precedence.
  - Next in precedence comes the concatenation (.) or dot operator.
  - Finally, unions (+) are grouped with their operands.
  - For example:
  - The expression 10*+0 is grouped (1(0)*)+0.

Regular Language

- The regular languages are those languages that can be constructed from the three set operations:
  - Union
  - Concatenation
  - Kleen star.
- Let $\sum$ be an alphabet. The class of "regular languages" over $\sum$ is defined inductively as follows:
  - ϕ is a regular language so as empty string $\{\varepsilon\}$ is regular language.
  - For each symbol $\boldsymbol{\sigma}$ ε Σ. $\{\sigma\}$ is a regular language.
  - If L1, L2, L3 are languages, then so is $L_1 \cup L2 \cup L3 \cup \ldots\ldots$ are also regular.

Regular Language

- If L1, L2 languages, then so is L_1 o L2 o L3 o ..... are also regular
- If L is a regular language, then so is L*.
- Nothing else is a regular language unless its construction follows from the above rules.

Algebraic Law for RL

- Commutative law for union: $L + M = M + L$
- Associative law for union: $(L + M) + N = L + (M + N)$
- Associative law for concatenation: $(LM)N = L(MN)$, ***Note that there is no commutative law for Concatenation, i.e. $LM \neq ML$***
- The identity for union is: $L + \phi = \phi + L = L$
- The identity for concatenation is: $L.\ \varepsilon = \varepsilon.\ L = L$
- Left distributive law: $L(M + N) = LM + LN$
- Right distributive law: $(M + N)L = LM + LN$
- Idempotent law: $L + L = L$

Algebraic Law for RE

- ***Laws Involving Closure***
  - $(L^*)^* = L^*$ – i.e. closing an already closed expression does not change the language
  - $\phi^* = \varepsilon$
  - $\varepsilon^* = \varepsilon$
  - $L^+ = L.\ L^* = L^*L$

Some Questions

- Write a regular expression for the language. L: {w $\in$ {a, b}*: w has even number of 'a's followed by odd number of 'b' s
- Write a regular expression for the set of strings over ∑{0, 1 } with exactly two 0's.
- Write regular expressions for the language which generates strings of even length over the alphabet ∑{a, b}.
- Describe as simply as possible in English the language corresponding to the regular expression $\mathbf{a^*b(a^*ba^*b)^*a^*}$
- Write a regular expression for the language in which strings start and end with same symbol over alphabet ∑{a, b }.
- Write a regular expression for the language in which strings start and end with different symbol over alphabet ∑{a, b }.

Some Questions

Q1. Answer: (aa)*(bb)*b

Q2. Answer: 1* 0 1*0 1*

Q3. Answer: (aa+ab+ba+bb)*

End of Chapter 1

Thank You !!!!