GURU GOBIND SINGH
INDRAPRASTHA UNIVERSITY
Paper Code: BCA 102

Paper ID: 20102

Paper: Mathematics – I

UNIT I

SETS: Sets, Subsets, Equal Sets Universal Sets, Finite and Infinite Sets, Operation on Sets, Union, Intersection and Complements of Sets, Cartesian Product, Cardinality of Set, Simple Applications.

RELATIONS AND FUNCTIONS: Properties of Relations, Equivalence Relation, Partial Order Relation Function: Domain and Range, Onto, Into and One to One Functions, Composite and Inverse Functions, Hashing functions, Recursive function.

UNIT – II

PARTIAL ORDER RELATIONS AND LATTICES: Partial Order Sets, Representation of POSETS using Hasse diagram, Chains, Maximal and Minimal Point, Glb, lub, Lattices & Algebric Systems, Principle of Duality, Basic Properties, Sub lattices, Distributed & Complemented Lattices.

UNIT-III

Graphs: types and operations(bipartite graph. Subgraph, distance of a graph, cut-edges & cut vertices, isomorphic and homomorphic graphs), degree of graphs, adjacent and incidence matrices, path circuit(Floyd's and Warshall algorithms), hamiltonian graph, graph colouring.

UNIT – IV

Propositional Logic: Proposition, First order logic, Basic logical operation, truth tables, tautologies, contradictions, Algebra of Proposition, logical implications, logical equivalence, predicates, Universal and existential quantifiers.

Unit -1

Definition OF SETS

A set is a well defined collection of distinct objects. The objects that make up a set (also known as the elements or members of a set) can be anything: numbers, people, letters of the alphabet,

other sets, and so on. Georg Cantor, the founder of set theory, gave the following definition of a set at the beginning of his *Beiträge zur Begründung der transfinite Mengenlehre*

A set is a gathering together into a whole of definite, distinct objects of our perception or of our thought ï which are called elements of the set.

Sets are conventionally denoted with capital letters. Sets *A* and *B* are equal if and only if they have precisely the same elements

As discussed below, the definition given above turned out to be inadequate for formal mathematics; instead, the notion of a "set" is taken as an undefined primitive in axiomatic set theory, and its properties are defined by the Zermeloï Fraenkel axioms. The most basic properties are that a set "has" elements, and that two sets are equal (one and the same) if and only if every element of one is an element of the other.

Describing sets

There are two ways of describing, or specifying the members of, a set. One way is by intensional definition, using a rule or semantic description:

A is the set whose members are the first four positive integers.
B is the set of colors of the French flag.

The second way is by extension ï that is, listing each member of the set. An extensional definition is denoted by enclosing the list of members in curly brackets:

C = {4, 2, 1, 3}
D = {blue, white, red}.

Every element of a set must be unique; no two members may be identical. (A multi set is a generalized concept of a set that relaxes this criterion.) All set operations preserve this property. The order in which the elements of a set or multi set are listed is irrelevant (unlike for a sequence or tuple). Combining these two ideas into an example

{6, 11} = {11, 6} = {11, 6, 6, 11}

because the extensional specification means merely that each of the elements listed is a member of the set.

For sets with many elements, the enumeration of members can be abbreviated. For instance, the set of the first thousand positive integers may be specified extensionally as:

{1, 2, 3, ..., 1000},

where the ellipsis ("...") indicates that the list continues in the obvious way. Ellipses may also be used where sets have infinitely many members. Thus the set of positive even numbers can be written as {2, 4, 6, 8, ...}.

The notation with braces may also be used in an intentional specification of a set. In this usage, the braces have the meaning "the set of all ...". So, E = {playing card suits} is the set whose four members are ♠, ♦, ♥ and ♣. A more general form of this is set-builder notation, through which, for instance, the set F of the twenty smallest integers that are four less than perfect squares can be denoted:

$$F = \{n^2 - 4 : n \text{ is an integer; and } 0 \leq n \leq 19\}.$$

In this notation, the colon (":") means "such that", and the description can be interpreted as "F is the set of all numbers of the form $n^2 - 4$, such that n is a whole number in the range from 0 to 19 inclusive." Sometimes the vertical bar ("|") is used instead of the colon.

One often has the choice of specifying a set intensionally or extensionally. In the examples above, for instance, $A = C$ and $B = D$.

Membership

The key relation between sets is *membership* ∈ when one set is an element of another. If a is a member of B, this is denoted $a \in B$, while if c is not a member of B then $c \notin B$. For example, with respect to the sets A = {1,2,3,4}, B = {blue, white, red}, and $F = \{n^2 - 4 : n \text{ is an integer; and } 0 \leq n \leq 19\}$ defined above,

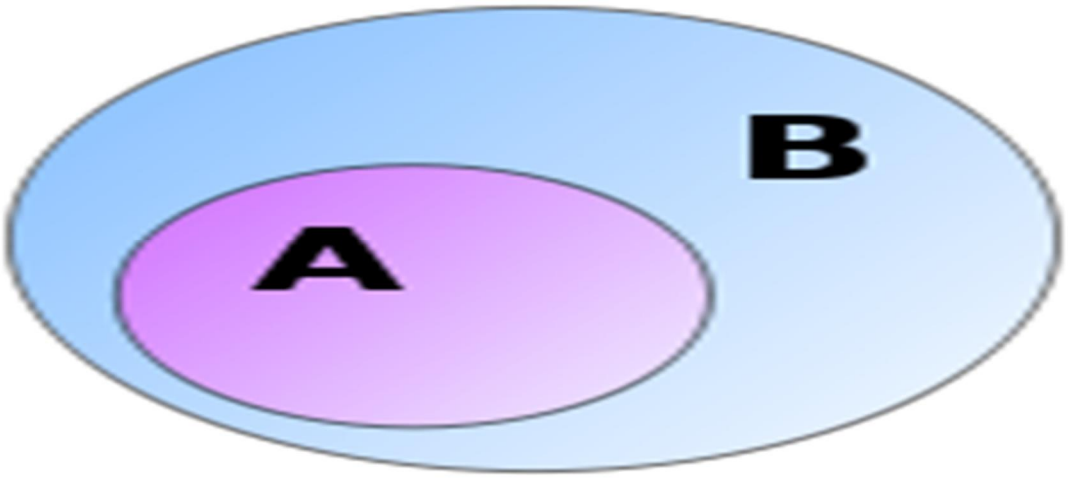
$4 \in A$ and $12 \in F$; but
$9 \notin F$ and green $\notin B$.

Subsets

If every member of set A is also a member of set B, then A is said to be a *subset* of B, written $A \subseteq B$ (also pronounced *A is contained in B*). Equivalently, we can write $B \supseteq A$, read as *B is a superset of A*, *B includes A*, or *B contains A*. The relationship between sets established by ⊆ is called *inclusion* or *containment*.

If A is a subset of, but not equal to, B, then A is called a *proper subset* of B, written $A \subsetneq B$ (*A is a proper subset of B*) or $B \supsetneq A$ (*B is a proper superset of A*).

Note that the expressions $A \subset B$ and $B \supset A$ are used differently by different authors; some authors use them to mean the same as $A \subseteq B$ (respectively $B \supseteq A$), whereas other use them to mean the same as $A \subsetneq B$ (respectively $B \supsetneq A$).



A is a **subset** of *B*

Example:

- The set of all men is a proper subset of the set of all people.
- $\{1, 3\} \subsetneq \{1, 2, 3, 4\}$.
- $\{1, 2, 3, 4\} \subseteq \{1, 2, 3, 4\}$.

The empty set is a subset of every set and every set is a subset of itself:

- $\emptyset \subseteq A$.
- $A \subseteq A$.

An obvious but useful identity, which can often be used to show that two seemingly different sets are equal:

- $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$.

A partition of a set *S* is a set of nonempty subsets of *S* such that every element *x* in *S* is in exactly one of these subsets.

Power sets

The power set of a set *S* is the set of all subsets of *S*, including *S* itself and the empty set. For example, the power set of the set {1, 2, 3} is {{1, 2, 3}, {1, 2}, {1, 3}, {2, 3}, {1}, {2}, {3}, ∅}. The power set of a set *S* is usually written as $P(S)$.

The power set of a finite set with *n* elements has 2^n elements. This relationship is one of the reasons for the terminology *power set*. For example, the set {1, 2, 3} contains three elements, and the power set shown above contains $2^3 = 8$ elements.

The power set of an infinite (either countable or uncountable) set is always uncountable. Moreover, the power set of a set is always strictly "bigger" than the original set in the sense that there is no way to pair the elements of a set *S* with the elements of its power set *P*(*S*) such that every element of *S* set is paired with exactly one element of *P*(*S*), and every element of *P*(*S*) is paired with exactly one element of *S*. (There is never a bijection from *S* onto *P*(*S*).)

Every partition of a set *S* is a subset of the power set of *S*.

Cardinality

The cardinality $|S|$ of a set *S* is "the number of members of *S*." For example, if *B* = {*blue, white, red*}, $|B|$ = 3.

There is a unique set with no members and zero cardinality, which is called the *empty set* (or the *null set*) and is denoted by the symbol ∅ (other notations are used; see empty set). For example, the set of all three-sided squares has zero members and thus is the empty set. Though it may seem trivial, the empty set, like the number zero, is important in mathematics; indeed, the existence of this set is one of the fundamental concepts of axiomatic set theory.

Some sets have infinite cardinality. The set **N** of natural numbers, for instance, is infinite. Some infinite cardinalities are greater than others. For instance, the set of real numbers has greater cardinality than the set of natural numbers. However, it can be shown that the cardinality of (which is to say, the number of points on) a straight line is the same as the cardinality of any segment of that line, of the entire plane, and indeed of any finite-dimensional Euclidean space.

Special sets

There are some sets that hold great mathematical importance and are referred to with such regularity that they have acquired special names and notational conventions to identify them. One of these is the empty set, denoted {} or ∅. Another is the unit set {x}, which contains exactly one element, namely x. Many of these sets are represented using blackboard bold or bold typeface. Special sets of numbers include:

- **P** or ℙ, denoting the set of all primes: **P** = {2, 3, 5, 7, 11, 13, 17, ...}.
- **N** or ℕ, denoting the set of all natural numbers: **N** = {1, 2, 3, . . .} (sometimes defined containing 0).
- **Z** or ℤ, denoting the set of all integers (whether positive, negative or zero): **Z** = {..., ī 2, ī 1, 0, 1, 2, ...}.
- **Q** or ℚ, denoting the set of all rational numbers (that is, the set of all proper and improper fractions): **Q** = {*a*/*b* : *a*, *b* ∈ **Z**, *b* Í 0}. For example, 1/4 ∈ **Q** and 11/6 ∈ **Q**. All integers are in this set since every integer *a* can be expressed as the fraction *a*/1 (**Z** ⊊ **Q**).
- **R** or ℝ, denoting the set of all real numbers. This set includes all rational numbers, together with all irrational numbers (that is, numbers that cannot be rewritten as fractions, such as ã2, as well as transcendental numbers such as ´ , *e* and numbers that cannot be defined).
- **C** or ℂ, denoting the set of all complex numbers: **C** = {*a* + *bi* : *a*, *b* ∈ **R**}. For example, 1 + 2*i* ∈ **C**.
- **H** or ℍ, denoting the set of all quaternions: **H** = {*a* + *bi* + *cj* + *dk* : *a*, *b*, *c*, *d* ∈ **R**}. For example, 1 + *i* + 2*j* ī *k* ∈ **H**.

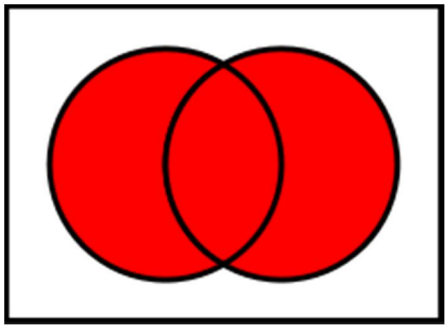
Positive and negative sets are denoted by a superscript - or +, for example: $\mathbb{Q}^+$ represents the set of positive rational numbers.

Each of the above sets of numbers has an infinite number of elements, and each can be considered to be a proper subset of the sets listed below it. The primes are used less frequently than the others outside of number theory and related fields.

Basic operations

There are several fundamental operations for constructing new sets from given sets.

Unions



The **union** of A and B, denoted $A \cup B$

Two sets can be "added" together. The *union* of A and B, denoted by $A \cup B$, is the set of all things that are members of either A or B.

Examples:

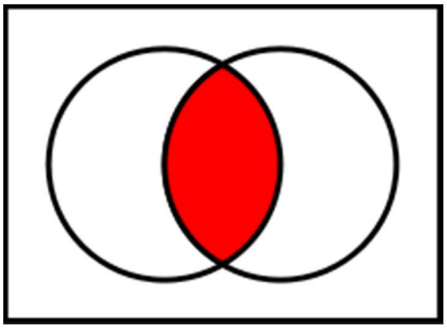
- $\{1, 2\} \cup \{1, 2\} = \{1, 2\}$.
- $\{1, 2\} \cup \{2, 3\} = \{1, 2, 3\}$.

Some basic properties of unions:

- $A \cup B = B \cup A$.
- $A \cup (B \cup C) = (A \cup B) \cup C$.
- $A \subseteq (A \cup B)$.
- $A \cup A = A$.
- $A \cup \emptyset = A$.
- $A \subseteq B$ if and only if $A \cup B = B$.

Intersections

A new set can also be constructed by determining which members two sets have "in common". The *intersection* of A and B, denoted by $A \cap B$, is the set of all things that are members of both A and B. If $A \cap B = \emptyset$, then A and B are said to be *disjoint*.



The **intersection** of A and B, denoted $A \cap B$.

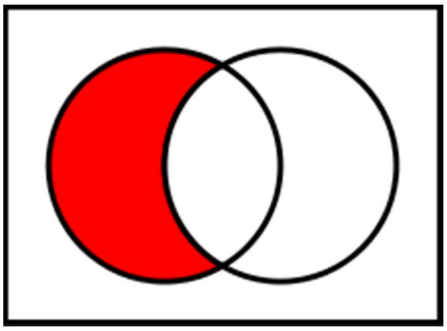
Examples:

- $\{1, 2\} \cap \{1, 2\} = \{1, 2\}$.
- $\{1, 2\} \cap \{2, 3\} = \{2\}$.

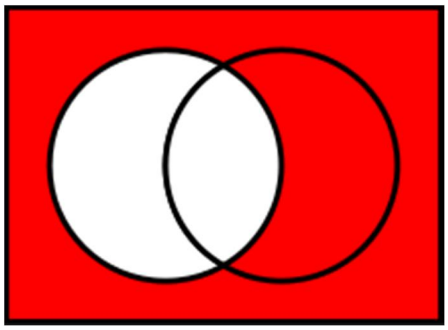
Some basic properties of intersections:

- $A \cap B = B \cap A$.
- $A \cap (B \cap C) = (A \cap B) \cap C$.
- $A \cap B \subseteq A$.
- $A \cap A = A$.
- $A \cap \emptyset = \emptyset$.
- $A \subseteq B$ if and only if $A \cap B = A$.

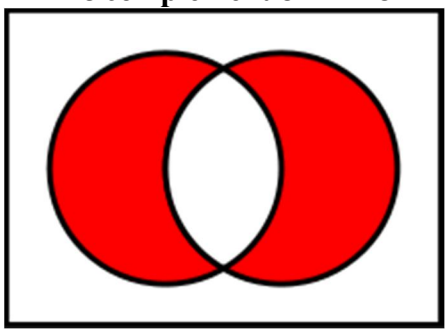
Complements



The **relative complement**
of B in A



The **complement** of *A* in *U*



The **symmetric difference** of *A* and *B*

Two sets can also be "subtracted". The *relative complement* of *B* in *A* (also called the *set-theoretic difference* of *A* and *B*), denoted by *A* \ *B* (or *A* ī *B*), is the set of all elements that are members of *A* but not members of *B*. Note that it is valid to "subtract" members of a set that are not in the set, such as removing the element *green* from the set {1, 2, 3}; doing so has no effect.

In certain settings all sets under discussion are considered to be subsets of a given universal set *U*. In such cases, *U* \ *A* is called the *absolute complement* or simply *complements* of *A*, and is denoted by *A*Nj

Examples:

- {1, 2} \ {1, 2} = ∅.
- {1, 2, 3, 4} \ {1, 3} = {2, 4}.
- If *U* is the set of integers, *E* is the set of even integers, and *O* is the set of odd integers, then *U* \ *E* = *E*N╪ *O*.

Some basic properties of complements:

- *A* \ *B* Í *B* \ *A* for *A* Í *B*.
- *A* ∪ *A*N╪ *U*.
- *A* ž *A*N╪ ∅.
- (*A*N)N╪ *A*.

- $A \setminus A = \emptyset$.
- $U \not\subseteq \emptyset$ and $\emptyset \not\subseteq U$.
- $A \setminus B = A \cap B^c$

An extension of the complement is the symmetric difference, defined for sets A, B as

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

For example, the symmetric difference of {7,8,9,10} and {9,10,11,12} is the set {7,8,11,12}.

Cartesian product

A new set can be constructed by associating every element of one set with every element of another set. The *Cartesian product* of two sets A and B, denoted by $A \times B$ is the set of all ordered pairs (a, b) such that a is a member of A and b is a member of B.

Examples:

- $\{1, 2\} \times \{\text{red, white}\} = \{(1, \text{red}), (1, \text{white}), (2, \text{red}), (2, \text{white})\}$.
- $\{1, 2\} \times \{\text{red, white, green}\} = \{(1, \text{red}), (1, \text{white}), (1, \text{green}), (2, \text{red}), (2, \text{white}), (2, \text{green})\}$.
- $\{1, 2\} \times \{1, 2\} = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$.

Some basic properties of cartesian products:

- $A \times \underline{\emptyset} = \emptyset$.
- $A \times (B \cup C) = (A \times B) \cup (A \times C)$.
- $(A \cup B) \times C = (A \times C) \cup (B \times C)$.

Let A and B be finite sets. Then

- $|A \times B| = |B \times A| = |A| \times |B|$.

Applications

Set theory is seen as the foundation from which virtually all of mathematics can be derived. For example, structures in abstract algebra, such as groups, fields and rings, are sets closed under one or more operations.

One of the main applications of naive set theory is constructing relations. A relation from a domain A to a codomain B is a subset of the Cartesian product $A \times B$. Given this concept, we are

quick to see that the set F of all ordered pairs (x, x^2), where x is real, is quite familiar. It has a domain set **R** and a codomain set that is also **R**, because the set of all squares is subset of the set of all reals. If placed in functional notation, this relation becomes $f(x) = x^2$. The reason these two are equivalent is for any given value, y that the function is defined for, its corresponding ordered pair, (y, y^2) is a member of the set F.

Definition of a relation.
We still have not given a formal definition of a relation between sets X and Y. In fact the above way of thinking about relations is easily formalized, as was suggested in class by Adam Osborne: namely, we can think of a relation R as a function from X×Y to the two-element set {TRUE, FALSE}. In other words, for (x, y) 2 X×Y,

we say that xRy if and only if f((x, y)) = TRUE.

Properties of relations.

Let X be a set. We now consider various properties that a relation R on X ï i.e., R _ X × X may or may not possess.

(R1) Reflexivity: for all x 2 X, (x, x) 2 R.In other words, each element of X bears relation R to itself. Another way to say this is that the relation R contains the equality relation. Exercise X.X: Go back and decide which of the relations in Examples X.X above are reflexive. For instance, set membership is certainly not necessarily reflexive: 1 62 1 (and in more formal treatments of set theory, a set containing itself is usually explicitly prohibited), but _ is reflexive

.(R2) Symmetry: for all x, y 2 X, if (x, y) 2 R, then (y, x) 2 R .Again, this has a geometric interpretation in terms of symmetry across the diagonal y = x. For instance, the relation associated to the function $y = 1/x$ is symmetric since interchanging x and y changes nothing, whereas the relation associated to the function $y = x_2$ is not. (Looking ahead a bit, a function y = f(x) is symmetric iff it coincides with its own inverse function.)Exercise X.X: Which of the relations in Examples X.X above are symmetric?

(R3) Anti-Symmetry: for all x, y 2 X, if (x, y) 2 R and (y, x) 2 R, then x = y.
For instance, _ satisfies anti-symmetry.
Exercise X.X: Which of the relations in Examples X.X above are anti-symmetric?

(R4) Transitivity: for all x, y, z 2 X, if (x, y) 2 R and (y, z) 2 R, then (x, z) 2 R.
For instance, ñbeing a parent ofòis not transitive, but ñbeing an ancestor ofò
is transitive.

Definition: An equivalence relation on a set X is a relation on X which is reflexive, symmetric and transitive.

Examples of equivalence relations.

Let n be a positive integer. Then there is a natural partition of Z into n parts which generalizes the partition into even and odd. Namely, we put Y_1 = {. . . ,ī 2n,ī n, 0, n, 2n, . . .} = {kn | k 2 Z} the set of all multiples of n,Y_2 = {. . . ,ī 2n + 1,ī n + 1, 1, n + 1, 2n + 1 . . .} = {kn + 1 | k 2 Z}, and similarly, for any 0 _ d _ n ī 1, we put Y_d = {. . . ,ī 2n + d,ī n + d, d, n + d, 2n + d . . .} = {kn + d | kinZ}. That is, Y_d is the set of all integers which, upon division by n, leave a remainder of d. Earlier we showed that the remainder upon division by n is a well-defined integer in the range 0 _ d < n. Here by ñwell-definedò I mean that for 0 _ d_1 6= d_2 < n, the sets Y_{d_1} and Y_{d_2} are disjoint. Recall why this is true: if not, there exist k_1, k_2 such that k_1n + d_1 = k_2n + d_2, so d_1 ī d_2 = (k_2 ī k_1)n, so d_1 ī d_2 is a multiple of n. But ī n < d_1 ī d_2 < n, so the only multiple of n it could possibly be is 0, i.e., d_1 = d_2. It is clear that each Y_d is nonempty and that their union is all of Z, so $\{Y_d\}_{n\bar{\imath}\,1\ d=0}$ gives a partition of Z. The corresponding equivalence relation is called congruence modulo n, and written as follows: x _ y (mod n). What this means is that x and y leave the same remainder upon division by n.

definition

A (non-strict) **partial order** is a binary relation "Ò' over a set *P* which is reflexive, anti symmetric, and transitive, i.e., which satisfies for all *a*, *b*, and *c* in *P*:

- $a \leq a$ (reflexivity);
- if $a \leq b$ and $b \leq a$ then $a = b$ (anti symmetry);
- if $a \leq b$ and $b \leq c$ then $a \leq c$ (transitivity).

In other words, a partial order is an anti symmetric preorder.

A set with a partial order is called a **partially ordered set** (also called a **poset**). The term *ordered set* is sometimes also used for posets, as long as it is clear from the context that no other kinds of orders are meant. In particular, totally ordered sets can also be referred to as "ordered sets", especially in areas where these structures are more common than posets.

For *a, b*, elements of a partially ordered set *P*, if $a \leq b$ or $b \leq a$, then *a* and *b* are **comparable**. Otherwise they are **incomparable**. In the figure on top-right, e.g. {x} and {x,y,z} are comparable, while {x} and {y} are not. A partial order under which every pair of elements is comparable is called a **total order** or **linear order**; a totally ordered set is also called a **chain** (e.g., the natural numbers with their standard order). A subset of a poset in which no two distinct elements are comparable is called an **antichain** (e.g. the set of singletons {{x}, {y}, {z}} in the top-right figure). An element *a* is said to be **covered** by another element *b*, written *a*<:*b*, if *a* is strictly less than *b* and no third element *c* fits between them; formally: if both *a*Ò*b* and *a*Í *b* are true, and *a*Ò*c*Ò*b* is false for each *c* with *a*Í *c*Í *b*. A more concise definition will be given below using the strict order corresponding to "Ò'. For example, {x} is covered by {x,z} in the top-right figure, but not by {x,y,z}.

Standard examples of posets arising in mathematics include:

- The real numbers ordered by the standard *less-than-or-equal* relation ≤ (a totally ordered set as well).

- The set of subsets of a given set (its power set) ordered by inclusion (see the figure on top-right). Similarly, the set of sequences ordered by subsequence, and the set of strings ordered by substring.

- The set of natural numbers equipped with the relation of divisibility.

- The vertex set of a directed acyclic graph ordered by reach ability.

- The set of subspaces of a vector space ordered by inclusion.

- For a partially ordered set *P*, the sequence space containing all sequences of elements from *P*, where sequence *a* precedes sequence *b* if every item in *a* precedes the corresponding item in *b*. Formally, $(a_n)_{n\in\mathbb{N}} \leq (b_n)_{n\in\mathbb{N}}$ if and only if $a_n \leq b_n$ for all *n* in $\mathbb{N}$.

- For a set *X* and a partially ordered set *P*, the function space containing all functions from *X* to *P*, where $f \leq g$ if and only if $f(x) \leq g(x)$ for all *x* in *X*.

- A fence, a partially ordered set defined by an alternating sequence of order relations $a < b > c < d$ ...

Function

Consider the relation
f : {(a, 1), (b, 2), (c, 3), (d, 5)}
In this relation we see that each element of A has a unique image in *B* This relation *f* from set *A* to *B* where every element of *A* has a unique image in *B* is defined as a function from *A* to *B*. So we observe that ***in a function no two ordered pairs have the same first element.***

Domain and Range:-
We also see that ∃ an element ∈ B, i.e., 4 which does not have its preimage in A. Thus here:
(i) the set B will be termed as co-domain and
(ii) the set {1, 2, 3, 5} is called the range.
From the above we can conclude that ***range is a subset of co-domain.*** Symbolically, this function can be written as
$f : A \to B$ or $A \xrightarrow{f} B$

Example

Which of the following relations are functions from A to B. Write their domain and range. If it is not a function give reason?

(a) $\{ (1, -2),(3,7),(4, -6),(8,1) \}$, $A = \{1,3,4,8\}$, $B = \{-2,7, -6,1,2\}$

(b) $\{ (1,0),(1 - 1),(2,3),(4,10) \}$, $A = \{1,2,4\}$, $B = \{0, -1,3,10\}$

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

(d) $\{(2,4),(3,9),(4,16),(5,25),(6,36\}$, $A = \{2,3,4,5,6\}$, $B = \{4,9,16,25,36\}$

(e) $\{(1,-1),(2,-2),(3,-3),(4,-4),(5,-5)\}$,$A = \{0,1,2,3,4,5\}$,

$B = \{-1,-2,-3,-4,-5\}$

Solution :

(a) It is a function.

Domain= $\{1,3,4,8\}$, Range = $\{-2,7,-6,1\}$

(b) It is not a function. Because Ist two ordered pairs have same first elements.

(c) It is not a function.

Domain= $\{a,b,c,d\} \neq A$, Range = $\{b, c\}$

(d) It is a function.

Domain = $\{2,3,4,5,6\}$, Range = $\{4,9,16,25,36\}$

(e) It is not a function .

Domain = $\{1,2,3,4,5\} \neq A$, Range = $\{-1,-2,-3,-4,-5\}$

Types of functions :-

One-to-one:- Let *f* be a function from *A* to *B*. If every element of the set *B* is the image of at least one element of the set *A* i.e. if there is no unpaired element in the set *B* then we say that the function f maps the set *A* onto the set *B*. Otherwise we say that the function maps the set *A* into the set *B*. Functions for which each element of the set *A* is mapped to a different element of the set *B* are said to be ***one-to-one.***

Many-to-one.:- A function can map more than one element of the set A to the same element of the set B. Such a type of function is said to be ***many-to-one.***

Reciprocal Function/Inverse function:-

Functions of the type $y = 1/x$, $x \neq 0$, called a reciprocal function.

Composite function:- Consider the two functions given below:

$y = 2x + 1$, $x \in \{1,2,3\}$

$z = y + 1$, $y \in \{3,5,7\}$

Then *z* is the composition of two functions *x* and *y* because *z* is defined in terms of *y* and *y* in terms of *x*.

Graphically one can represent this as given below :

HASHING FUNCTION

To save space and time, each record stored in a computer is assigned an address (memory location) in the computer's memory. The task of assigning the address is performed by the Hashing function (or Hash function) $H : K \to L$, which maps the set K of keys to the set L of memory addresses. Thus a Hashing function provides means of identifying records stored in a table. The function H should be one-to-one. In fact, if $k_1 \neq k_2$ implies $H(k_1) = H(k_2)$, then two keys will have same address and we say that *collision* occurs. To resolve collisions, the following methods are used to define the hash function.

1. **Division Method.** In this method, we restrict the number of addresses to a fixed number (generally a prime) say m and define the hash function $H : K \to L$ by

$$H(k) = k \pmod{m},\ k \in K,$$

where k (mod m) denotes the remainder when k is divided by m.

2. **Midsquare Method.** As the name suggest, we square the key in this method and define hash function $H : K \to L$ by $H(k) = l$, where l is obtained by deleting digits from both ends of k^2.
3. **Folding Method.** In this method the key k is partitioned into a number of parts, where each part, except possibly the last, has the same numbers of digits as the required memory address. Thus, if $k = k_1 + k_2 + \text{é} + k_n$, then the hash function $H : K \to L$ is defined by

$H(k) = k_1 + k_2 + \text{é} + k_n$, where the last carry has been ignored.

Definition of recursive functions:- The class of **recursive functions** is defined as follows:The functions s and z are recursive, and so are all projections ${}_ip^k$. Functions built from recursive ones by using composition Cn or primitive recursion Pr are\ also recursive.Functions built from recursive ones by\minimization Mn are also recursive

UNIT -2 (PARTIAL ORDER RELATIONS AND LATTICES)

Partial Order Relations on a Lattice:-

A partial order relation on a lattice (L) follows as a consequence of the axioms for the binary operations ∨ and ∧.

PARTIALLY ORDERED SETS

A relation R on a set X is said to be anti symmetric if a R b and b R a imply a = b. Relation R on a set X is called a partial order relation if it is reflexive, anti-symmetric and transitive. A set X with the partial order R is called a partially ordered set or poset and is denoted by (X, R)

EXAMPLE

Let $\tilde{A}$ be a collection of subsets of a set S. The relation $\subseteq$ of set inclusion is a partial order relation on $\tilde{A}$. In fact, if $A, B, C \in \tilde{A}$, then,

$A \subseteq A$, that is, A is a subset of itself which is true.

If $A \subseteq B$, $B \subseteq A$, then $A = B$

If $A \subseteq B$, $B \subseteq C$, then A is a subset of C, that is, $A \subseteq C$.

HASSE DIAGRAM

Let A be a finite set. By the theorem proved above, the digraph of a partial order on A has only cycles of length 1. In fact, since a partial order is reflexive, every vertex in the digraph of the partial order is contained in a cycle of length 1. To simplify the matter, we shall delete all such cycles from the digraph.

We also eliminate all edges that are implied by transitivity property. Thus, if $a \leq b$, $b \leq c$, it follows that $a \leq c$. In this case, we omit the edge form a to c. We also agree to draw the digraph of partial order with all edges pointing upward, omit the arrows and to replace then the circles by dots

"The diagram of a partial order obtained from its digraph by omitting cycles of length 1, the edges implied by transitivity and the arrows (after arranging them pointing upward) is called Hasse diagram of the partial order of the poset".

EXAMPLE

Let A = {1, 2, 3, 4, 12}. Consider the partial order of divisibility on A. That is, if a and b are in A, $a \leq b$ if and only if $a \mid b$

Therefore, the Hasse diagram of the poset $(A, \leq)$ is as shown in Figure 1.18.

EXAMPLE

Let S = {a, b, c} and $\tilde{A} = P(S)$ (power set of S).

Consider the partial order of set inclusion ($\subseteq$). We note that

$\tilde{A} = P(S) = \{ \varphi, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$

Then the Hasse diagram of the poset $(\tilde{A}, \subseteq)$

Hasse diagram of a finite linearly ordered set is always of the form and thus consists of simply one path. Hence diagram of a totally order set is a chain.

Hasse Diagram of Dual Poset

If (A, ≤) is a poset and (A, ≥) is the dual poset, then the Hasse diagram of (A, ≥) is just the Hasse diagram of (A, ≤) turned upside down.

For example, let A = {a, b, c, d, e, f} and let be the Hasse diagram of poset (A, ≤). Then the Hasse diagram of dual poset (A, ≥) is which can be constructed by turning the Hasse diagram of (A, ≤) upside down.

EXAMPLE Let A = {a, b, c, d, e}. Then the Hasse diagram defines a partial order on B in the natural way. That is, d ≤ b, d ≤ a, e ≤ c and so on.

EXAMPLE Let n be a positive integer and Dn denote the set of all divisors of n. Considering the partial order of divisibility in Dn, draw Hasse diagram D24, D30 and D36.

Solution.

We know that

D24 = {1, 2, 3, 4, 6, 8, 12, 24},

D30 = {1, 2, 3, 5, 6, 10, 15, 30},

D36 = {1, 2, 3, 4, 6, 9, 12, 18, 36}.

Therefore, the Hasse diagram of D24, D30 and D36

{5}, {3, 2}, {2, 2, 1}, {1, 1, 1, 1, 1}, {4, 1}, {3, 1, 1}, {2, 1, 1, 1}.

We order the partitions of an integer m as follows:

A partition P1 precedes a partition P2 if the integers in P1 can be added to obtain integers in P2 or we can say that if the integers in P2 can be further subdivided to obtain the integers in P1. For example, {1, 1, 1, 1, 1} precedes {2, 1, 1, 1}. On the other hand, {3, 1, 1} and {2, 2, 1} are non-comparable.

The Hasse diagram of the partition of m = 5 is

Let A be a (non-empty) linearly ordered alphabet. Then Kleene closure of A consists of all words w on A and is denoted by A*.

Also then |w| denotes the length of w.

We have following two order relations on A*.

Alphabetical (Lexicographical) order: In this order we have

λ < w, where λ is empty word and w is any non-empty word.

Suppose u = a u' and v = b v' are distinct non-empty words where a, b ∈ A and u', v' ∈ A*. Then,

u < v if a < b or if a = b but u' < v' Short-lex order: Here A* is ordered first by length and then alphabetically, that is, for any distinct words u, v, in A*, u < v if |u| < |v| or if |u| = |v| but u precedes v alphabetically. For example, "to" proceeds "and" since |to| = 2 but |and| = 3. Similarly, "an" precedes "to" since they have the same length but "an" precedes "to" alphabetically.

This order is also called free semi-group order.

Let A be a partially ordered set with respect to a relation ≤. An element a in A is called a maximal element of A if and only if for all b in A, either b ≤ a or b and a are not comparable.

An element a in A is called greatest element of A if and only if for all b in A, b ≤ a.

An element a in A is called minimal element of A if and only if for all b in A, either a ≤ b or b and a are not comparable.

An element a in A is called a least element of A if and only if for all b in A, a ≤ b.

A greatest element is maximal but a maximal element need not be greatest element. Similarly, a least element is minimal but a minimal element need not be a least element.

The elements a1, a2 and a3 are maximal elements of A, and the elements b1, b2 and b3 are minimal elements. Observe that since there is no line between b2 and b3 we can conclude that neither b3 ≤ b2 nor b2 ≤ b3 showing that b2 and b3 are not comparable.

Let A be the poset of non-negative real numbers with usual partial order ≤ (read as " less than or equal to "). Then 0 is the minimal element of A. There is no maximal element of A.

Let A be a finite non-empty poset with partial order ≤. Then A has at least one maximal element and at

.

Let (A, ≤) be a poset and B a subset of A. An element a ∈ A is called a least upper bound (supremum) of B if

a is an upper bound of B, that is, b ≤ a ∀ b ∈ B

a ≤ a' whenever a' is an upper bound of B.

An element a ∈ A is called a greatest lower bound (infimum) of B if

a is a lower bound of B, that is, a ≤ b ∀ b ∈ B

a' ≤ a whenever a' is a lower bound of B.

Further, upper bounds in the poset (A, ≤) correspond to lower bounds in the dual poset (A, ≥) and the lower bounds in (A, ≤) correspond to upper bound in (A, ≥).

Similar statements also hold for greatest lower bounds and least upper bounds.

Consider Example 1.69 above:

Since B1 = {a, b} has no lower bound, it has no greatest lower bound. However,

lub (B1) = c

Since the lower bounds of B2 = {c, d, e} are c, a and b, we have

glb (B2) = c

The upper bounds of B2 are f, g, h. Since f and g are not comparable, we conclude that B2 has no least upper bound.

(Here d and e are not upper bounds of {c, d, e} because d e ∈ B2 and e d∈B2)

EXAMPLE

Let A = {1, 2, 3, 4, 5, …, 11}be the poset whose Hasse diagram is given (Figure 1.32).

Figure Find lub and glb of B = {6, 7, 10}, if they exist.

Solution.

Exploring all upward paths from 6, 7 and 10 we find that lub (B) = 10. Similarly, by examining all downward paths from 6, 7 and 10, we find that glb (B) = 4.

EXAMPLE Let Dn denote the set of factors of a positive integer n partially ordered by divisibility. Then,

Let ≤ and ≤' be two partial order relations on a set A. Then ≤' is said to be compatible with ≤ if and only if

$a \leq b \Rightarrow a \leq' b$ for all $a, b \in A$.

The process of constructing a linear order (total order) which is compatible to a given partial order on a given set is called topological sorting.

The construction of a topological sorting for a general finite partially order set is based on the fact that any partially ordered set that is finite and non-empty has a minimal element.

To create a total order for a partially ordered set (A, ≤), we proceed as follows:

Pick any minimal element and make it number one. Let this element be a.

Consider A – {a}. It is a subset of A and so is partially ordered. If it is empty, stop the process. If not, pick a minimal element from it and call it element number 2. Let it be b.

Consider A – {{a}, {b}}. If this set is empty, stop the process. If not, pick a minimal element and call it number 3. Continue in this way until all the elements of the set have been used up.

We now give algorithm to construct a topological sorting for a relation ≤ defined on a non-empty finite set A.

Then, remove one of 4 and 18. If we remove 18, we get

Figure

total order : 3 ≤′ 2 ≤′ 6* ≤′ 18

Then,

A = (A – {3, 2, 6, 18})

Now minimal element is 4. We remove it and we get

A = A – {3, 2, 6, 18, 4} = {24}

total order : 3 ≤′ 2 ≤′ 6 ≤′ 18 ≤′ 4 ≤′ 24

(Hasse diagram of A – {d})

The minimal element of A – {d} is e and we put e in sort [2]. The Hasse diagram of A – {d, (Hasse diagram of A – {d, c}).

The minimal element of A – {d, e} is c and we put e in sort [3]. The Hasse diagram of A – {d, e, c} is as shown below .The minimal element of A – {d, e, c} are a and b. We pick b and put it in sort [4]. The Hasse diagram of A – {d, e, c, b} is shown below: (Hasse diagram of A – {d, e, c, b}).

The minimal element of A – {d, e, c, b} is a and we put it in sort [5]. The topological sorting of (A, ≤) is therefore (A, <), where total order : d < e < c < b < a and the Hasse diagram of (A, <) is as shown in the

Let N be the set of positive integers. Then the usual relation ≤ (read "less than or equal to") is a partial order on N.

Similarly, ≥ (read "greater than or equal to") is a partial order on N.

But the relation < (read "less than") is not a partial order on N. In fact, this relation is not reflexive.

EXAMPLE Let N be the set of positive integers. Then the relation of divisibility is a partial order on N. We say that "a divides b" written as a | b, if there exists an integer c such that a c = b. We note that for a, b, c ∈ N.

a | a

a | b, b | a ⇒ a = b

a | b, b | c ⇒ a | c.

Thus, relation of divisibility is a partial order on N.

EXAMPLE

The relation of divisibility is not a partial order on the set of integers. For example, 3 | −3, −3 | 3 but 3 ≠ −3, that is, the relation is not anti-symmetric and so cannot be partial order.

EXAMPLE

If R is a partial order on A, then R−1 (inverse rela□on) is also a par□al order.

We know that if R is a relation on A, then

R−1 = {(b, a) : (a, b) ∈ R}, a, b ∈ A.

Since R is a partial order relation,

a R a ∀ a ∈ A

a R b, b R a ⇒ a = b

a R b, b R c ⇒ a R c.

We observe that

(i) Since R is a relation, (a, a) ∈ R ∀ a ∈ A

⇒ (a, a) ∈ R−1

⇒ a R−1 a.

Thus the relation R−1 is reflexive.

(ii) If (b, a) ∈ R−1 and (a, b) ∈ R−1, then

(a, b) ∈ R and (b, a) ∈ R,

⇒ a R b and b R a,

⇒ a = b, since R is anti-symmetric.

Hence, R−1 is an□-symmetric.

(iii) If (b, a) ∈ R−1 and (c, b) ∈ R−1, then

(a, b) ∈ R and (b, c) ∈ R,

⇒ (a, c) ∈ R, since R is transitive

⇒ (c, a) ∈ R−1.

Thus (c, b) ∈ R−1 and (b, a) ∈ R−1 ⇒ (c, a) ∈ R−1 and so R−1 is transi□ve.

Hence R−1 is a par□al order.

The poset (A, R−1) is called the dual of the poset (A, R) and the par□al order R−1 is called the dual of the partial order R.

A relation R on a set A is said to be quasi order if

R is irreflexive, that is, (a, a) ∉ R for any a ∈ A

R is transitive, that is, a R b, b R c ⇒ a R c for a, b, c ∈ A.

Let (A, R) be a poset. The elements a and b of A are said to be comparable if a R b or b R a.

We know that the relation of divisibility is a partial order on the set of natural numbers. But we see that

3 7 and 7 3.

Thus, 3 and 7 are positive integers in N which are not comparable (In such a case we write 3 || 7).

If every pair of elements in a poset (A, R) is comparable, we say that A is linearly ordered (totally ordered or a chain). The partial order is then called linear order or total ordering relation. The number of elements in a chain is called the length of the chain.

Let A be a set with two or more elements and let ⊆ (set inclusion) be taken as the relation on the subsets of A. If a and b are two elements of A, then {a} and {b} are subsets of A but they are not comparable. Hence P(A) is not a chain. A subset of A is called Antichain if no two distinct elements in the subsets are related.

But, if we consider the subsets φ, {a} and A of A, then this collection (subsets { φ, {a}, A} of P(A)) is a chain because φ ⊆ {a} ⊆ A. Similarly, φ, {b} and A form a chain.

.

Let (a, b) R″ (a′, b′) and (a′, b′) R″ (a, b). Then, by definition,

a R a′, a′ R a in A (i)

b R' b', b' R' b in B. (ii)

Since (A, R) and (B, R') are posets, (i) and (ii) respectively imply

a = a'

and

b = b'.

Thus, (a, b) R" (a', b') and (a', b') R" (a, b) imply

(a, b) = (a', b').

Hence R" is anti-symmetric.

Let (a, b) R" (a', b') and (a', b') R" (a", b"),

where a, a', a" ∈ A and b, b', b" ∈ B. Then

a R a' and a' R a" (iii)

b R' b' and b' R' b". (iv)

By transitivity of R, (iii) gives

a R a",

while (iv) yields

b R' b".

Hence,

(a, b) R" (a", b").

Hence R" is transitive and so (A × B, R") is a poset.

The partial order R" defined on the Cartesian product A × B, as above, is called the Product Partial Order.

Definition 1.51

A relation R on a set A is called asymmetric if a R b and b R a do not both hold for any a, b ∈ A.

Definition 1.52

A transitive, asymmetric relation R is called a Strict Partial Ordering.

Theorem 1.27

If ≤ is a partial order of the set A, then a relation < defined by

a < b if a ≤ b and a ≠ b

is a strict partial order of A.

Proof. We shall show that < is transitive and asymmetric.

(i) Transitivity: Suppose that a < b and b < c. Then, by definition,

a ≤ b and b ≤ c, a ≠ b, b ≠ c. (1)

Since ≤ is partial order, it is transitive and so a ≤ c. It remains to show that a ≠ c. Suppose on the contrary that a = c. Then,

c ≤ b (using a ≤ b from (1)). (2)

From (1) and (2), we have b ≤ c and c ≤ b and so b = c which is contradiction. Hence,

a < b and b < c implies a < c.

Proving that < is transitive.

(ii) Asymmetry: Suppose that x < y and y < x both holds. Therefore,

$x \leq y$ and $y \leq x$.

Since $\leq$ is anti-symmetric, it follows that $x = y$, which contradicts $x < y$. Hence $x < y$ and $y < x$ cannot both hold. Thus $<$ is asymmetric.

Hence $<$ is strict partial order of A.

Remark 1.1 If $<$ is a strict partial order of A, then the relation $\leq$ defined by $x \leq y$ if $x < y$ or $x = y$ is a partial order of A (can be proved using the definitions).

Definition

A sequence of letters or other symbols, written without commas is called a string. Further,

A string of length p may be considered as an ordered p-tuple.

An infinite string such as abababab … may be regarded as the infinite sequence a, b, a, a, b, ab, …

If S is any set with a partial order relation, then the set of strings over S is denoted by S^*.

Definition Let $(A, \leq)$ and $(B, \leq)$ be chains (linearly ordered sets). Then the order relation (which is in fact totally ordered) $<$ on the Cartesian product $A \times B$ defined by

$(a, b) < (a', b')$ if $a < a'$ or if $a = a'$ and $b \leq b'$

is called Lexicographic order or Dictionary order.

EXAMPLE

Consider the plane $R2 = R \times R$. It is linearly ordered by lexicographic order. In fact, each vertical line has usual order (less than or equal to) and points on a line (e.g., $x = a1$ in Fig. 1.14) are less than any point on a line farther to the right (e.g. $x = a2$ in Fig. 1.14). Thus the point $p1\ (a1, b1) < p2\ (a2, b2)$ because $a1 < a2$. Further, $p2(a2, b2) < p3(a2, b3)$ because in this case $a2 = a2$ and $b2 \leq b3$.

Theorem :-The digraph (directed graph) of a partial order has no cycle of length greater than 1.

Proof. Suppose on the contrary that the digraph of the partial order $\leq$ on the set A contains a cycle of length $n \geq 2$. Then there exist distinct elements a1, a2, …, an such that

$a1 \leq a2, a2 \leq a3, \ldots, an-1 \leq an, an \leq a1.$

By the transitivity of partial order, used n − 1 mes, a1 ≤ an. Thus we have

a1 ≤ an and an ≤ a1.

Since ≤ is partial order, anti-symmetry implies a1 = an, which is a contradiction to the assumption that a1, a2… an are distinct. Hence the result.

Definition The Transitive closure of a relation R is the smallest transitive relation containing R. It is denoted by R∞.

We note that from vertex 1, we have paths to the vertices 2, 3, 4 and 1. Note that path from 1 to 1 proceeds from 1 to 2 to 1. Thus we see that the ordered pairs (1, 1), (1, 2), (1, 3) and (1, 4) are in R∞. Starting from vertex 2, we have paths to vertices 2, 1, 3 and 4 so the ordered pairs (2, 1), (2, 2), (2, 3) and (2, 4) are in R∞. The only other path is from vertex 3 to 4, so we have

R∞ = {(1, 1), (1, 2), (1, 3), (1, 4), (2, 1), (2, 2), (2, 3), (2, 4), (3, 4)}

A **Lattice** is an algebraic system $(L, \vee, \wedge)$ with two binary operations $\vee$ and $\wedge$, called **join** and **meet,** respectively, on a non-empty set L which satisfies the following axioms for $a, b, c \in L$:

1. **Commutative Law:**

$$a \vee b = b \wedge a \text{ and } a \wedge b = b \vee a.$$

2. **Associative Law:**

$$(a \vee b) \vee c = a \vee (b \vee c)$$

and

$$(a \wedge b) \wedge c = a \wedge (b \wedge c).$$

3. **Absorption Law:**

i. $a \vee (a \wedge b) = a$,

ii. $a \wedge (a \vee b) = a$.

We note that **Idempotent Law follows from axiom 3** above. In fact,

$a \vee a = a \vee [a \wedge (a \vee b)]$ using 3 (ii)
$= a$ using 3 (i).

The proof of $a \wedge a = a$ follows by the principle of duality.

LATTICE

Definition

A **lattice** is a partially ordered set $(L, \grave{O})$ in which every subset $\{a, b\}$ consisting of two elements has a least upper bound and a greatest lower bound.

We denote LUB($\{a, b\}$) by $a \vee b$ and call it **join** or **sum of** a **and** b. Similarly, we denote GLB ($\{a, b\}$) by $a \wedge b$ and call it **meet** or **product of** a **and** b.

Other symbols used are

LUB: $\oplus$, +, $\cup$,
GLB: *, ·, ž .

Thus **Lattice is** a mathematical structure with **two binary operations, join and meet.**

A totally ordered set is obviously a lattice but not all partially ordered sets are lattices.

EXAMPLE Let A be any set and $P(A)$ be its power set. The partially ordered set $(P(A), \subseteq)$ is a lattice in which the meet and join are the same as the operations ž and $\cup$, respectively. If A has single element, say a, then $P(A)=\{\blacklozenge, \{a\}\}$

PROPERTIES OF LATTICES

Let $(L, \grave{O})$ be a lattice and let $a, b, c \in L$. Then, from the definition of $\vee$ (join) and $\wedge$(meet) we have

i. $a \leq a \vee b$ and $b \leq a \vee b$; $a \vee b$ is an upper bound of a and b.

ii. If $a \le c$ and $b \le c$, then $a \vee b \le c$; $a \vee b$ is the least upper bound of a and b.
iii. $a \wedge b \le a$ and $a \wedge b \le b$; $a \wedge b$ is a lower bound of a and b.
iv. If $c \le a$ and $c \le b$, then $c \le a \wedge b$; $a \wedge b$ is the greatest lower bound of a and b.

Theorem

Let L be a lattice. Then for every a and b in L,

i. $a \vee b = b$ if and only if $a \le b$,
ii. $a \vee b = a$ if and only if $a \le b$,
iii. $a \wedge b = a$ if and only if $a \vee b = b$.

I. BOUNDED, COMPLEMENTED AND DISTRIBUTIVE LATTICES

ii. Let $(L, \vee, \wedge)$ be a lattice and let $S = \{a_1, a_2, \text{é} , a_n\}$ be a finite subset of L. Then,
iii.
iv. LUB of S is represented by $a_1 \vee a_2 \vee \ldots \vee a_n$,
GLB of S is represented by $a_1 \wedge a_2 \wedge \ldots \wedge a_n$.
v. **Definition** A lattice is called **complete** if each of its non-empty subsets has a least upper bound and a greatest lower bound.
vi. Obviously, every finite lattice is complete.
vii. Also, every complete lattice must have a least element, denoted by 0 and a greatest element, denoted by I.
viii. The least and greatest elements if exist are called **bound (units, universal bounds)** of the lattice.
ix. **Definition** A lattice L is said to be **bounded** if it has a greatest element I and a least element 0.
x. For the lattice $(L, \vee, \wedge)$ with $L = \{a_1, a_2, \text{é} , a_n\}$,

UNIT -3

DEFINITIONS AND BASIC CONCEPTS

Definition

A **graph $G = (V, E)$** is a mathematical structure consisting of two finite sets V and E. The elements of V are called **vertices (or nodes)** and the elements of E are called **edges**. Each edge is associated with a set consisting of **either one** or **two vertices** called its **endpoints**.

The correspondence from edges to endpoints is called **edge-endpoint function**. This function is generally denoted by ɔ. Due to this function, some authors denote graph by $G = (V, E, \gamma)$.

Definition

A graph consisting of one vertex and no edges is called a **trivial graph**.

Definition

A graph whose vertex and edge sets are empty is called a **null graph**.

Definition

An edge with just one endpoint is called a **loop** or a **self-loop**.

SPECIAL GRAPHS

Definition
A graph G is said to **simple** if it has no parallel edges or loops. In a simple graph, an edge with endpoints v and w is denoted by $\{v, w\}$.

Definition

For each integer n Ó 1, let D_n denote the graph with n **vertices** and **no edges**. Then D_n is called the **discrete graph on *n* vertices**.

Definition
Let n Ó 1 be an integer. Then a simple graph with n vertices in which there is an edge between each pair of distinct vertices is called the **complete graph** on n vertices. It is denoted by K_n.

For example, the complete graphs K_2, K_3 and K_4 are shown in

Definition
If each vertex of a graph G has the same degree as every other vertex, then G is called a **regular graph**.

SUBGRAPHS

Definition

A graph H is said to be a subgraph of a graph G if and only if every vertex in H is also a vertex in G, every edge in H is also an edge in G and every edge in H has the same endpoints as in G.

We may also say that *G* is a super graph of *H*
Definition

A sub graph H is said to be a **proper sub graph** of a graph G if vertex set V_H of H is a proper subset of the vertex set V_G of G or edge set E_H is a proper subset of the edge set E_G.

For example, the sub graphs in the above examples are proper sub graphs of the given graphs.

ISOMORPHISMS OF GRAPHS

We know that shape or length of an edge and its position in space are not part of specification of a graph. For example, the represent the same graph.

Definition

Let G and H be graphs with vertex sets $V(G)$ and $V(H)$ and edge sets $E(G)$ and $E(H)$, respectively. Then G **is said to isomorphic to** H if there exist one-to-one correspondences g: $V(G)$ Ÿ $V(H)$ and h: $E(G)$ Ÿ $E(H)$ such that for all $v \in V(G)$ and $e \in E(G)$,

$$v \text{ is an endpoint of } e \Leftrightarrow g(v) \text{ is an endpoint of } h(e).$$

Definition

The property of mapping endpoints to endpoints is called **preserving incidence** or **the continuity rule** for graph mappings.

As a consequence of this property, a self-loop must map to a self-loop.

Thus, two isomorphic graphs are same except for the labelling of their vertices and edges.

WALKS, PATHS AND CIRCUITS

Definition

In a graph G, **a walk from vertex** v_0 **to vertex** v_n is a finite alternating sequence $\{v_0, e_1, v_1, e_2,$ é $, v_{n-1}, e_n, v_n\}$ of vertices and edges such that v_{i-1} and v_i are the endpoints of e_i.

The **trivial walk** from a vertex v to v consists of the single vertex v.

Definition

In a graph G, a **path** from the vertex v_0 to the vertex v_n is a walk from v_0 to v_n that does not contain a repeated edge.

Thus a **path** from v_0 to v_n is a walk of the form

$$\{v_0, e_1, v_1, e_2, v_2, \ldots, v_{n-1}, e_n, v_n\},$$

where all the edges e_i are distinct.

Definition

In a graph, a **simple path** from v_0 to v_n is a path that does not contain a repeated vertex.

Thus a simple path is a walk of the form

$$\{v_0, e_1, v_1, e_2, v_2, \ldots, v_{i-1}, e_n, v_n\},$$

HAMILTONIAN CIRCUITS

Definition

A **Hamiltonian path** for a graph G is a sequence of adjacent vertices and distinct edges in which every vertex of G appears exactly once.

Definition

A **Hamiltonian circuit** for a graph G is a sequence of adjacent vertices and distinct edges in which every vertex of G appears exactly once, except for the first and the last which are the same.

Definition

A graph is called **Hamiltonian** if it admits a Hamiltonian circuit.

MATRIX REPRESENTATION OF GRAPHS

A graph can be represented inside a computer by using the adjacency matrix or the incidence matrix of the graph.

Definition

Let G be a graph with n ordered vertices v_1, v_2, é , v_n. Then the **adjacency matrix of** G is the $n \times n$ matrix $A(G) = (a_{ij})$ over the set of non-negative integers such that

$$a_{ij} = \text{the number of edges connecting } v_i \text{ and } v_j \text{ for all } i, j = 1, 2, \ldots, n.$$

We note that if G has no loop, then there is no edge joining v_i to v_i, $i = 1, 2$, é , n.Therefore, in this case, all the entries on the main diagonal will be 0.

Further, if G has no parallel edge, then the entries of $A(G)$ are either 0 or 1.

It may be noted that adjacent matrix of a graph is symmetric.

Conversely, given a $n \times n$ symmetric matrix $A(G) = (a_{ij})$ over the set of non-negative integers, we can associate with it a graph G, whose adjacency matrix is $A(G)$, by letting G have n vertices and joining v_i to vertex v_j by a_{ij} edges

COLOURING OF GRAPH

Definition

Let G be a graph. The assignment of colours to the vertices of G, one colour to each vertex, so that the adjacent vertices are assigned different colours is called **vertex colouring** or **colouring of the graph** G.

Definition

A graph G is ***n*-colourable** if there exists a colouring of G which uses n colours.

Definition

The minimum number of colours required to paint (colour) a graph G is called the **chromatic number of** G and is denoted by $\chi(G)$.

UNIT-4 Propositional Logic:

INTRODUCTION TO LOGIC

Logic is essentially the study of arguments. For example, someone may say, suppose A and B are true, can we conclude that C is true? Logic provides rules by which we can conclude that certain things are true given other things are true. Here is a simple example: A tells B that ñif it rains, then the grass will get wet. It is rainingò; B can then conclude that the grass is wet, if what A has told B is true. Logic provides a mechanism for showing arguments like this to be true or false.

The section starts by showing how to translate English sentences into a logical form, specifically into something called ñpropositionsò. In fact, our study of logic starts with *propositional calculus*.

Propositional calculus is the calculus of propositions and we plan to study propositions. Most of us may associate the term calculus with integrals and derivatives, but if we check out the definition of

calculus in the dictionary, we will see that calculus just means ña way of calculatingò, so differential calculus, for instance, is how to calculate with derivatives and integral calculus is how to calculate with integrals.

However, before going into how to translate English sentences into propositions, we are going to introduce Boolean expressions (that is, propositions), and then discuss about translation. Hence, we will see the same ideas in two different forms.

BOOLEAN EXPRESSIONS

Definition A Boolean variable is a variable that can either be assigned true or false. We have programmed in C++ and know about types such as integers, floats, and character pointers. However, C++ also has a Boolean type, as do Java and Pascal. We can declare variables to be of Boolean type, which means that they can only take on two values: true and false. Throughout the chapter, we shall generally use the letters, *p, q,* and *r* as Boolean variables. However, in some cases we will allow these letters to have subscripts. For example, p_0, q_{1492} and r_{1776} are all Boolean variables.

Definition A Boolean expression is either

1. a Boolean variable, or
2. it has the form ¬◂, where ◂ is a Boolean expression, or
3. it has the form (◂ * ψ), where ◂ and ψ are Boolean expressions and * is one of the following:

   ∧, ∨, Ÿ , or ź .

CONSTRUCTION OF BOOLEAN EXPRESSIONS

Suppose we are given a Boolean expression and asked to prove that it is a Boolean expression. How do we proceed? There are two different ways of doing it. The first is to build a Boolean expression from its constituent parts. Let us start off with an example. We want to show that $((p \wedge q) \vee \neg r)$ is a Boolean expression. To do so, we will take a bottom-up approach.

Expression	*Reason*
1. p	Boolean variable
2. q	Boolean variable
3. r	Boolean variable
4. $\neg r$	3, ¬◂
5. $(p \wedge q)$	1, 2, (◂ ∧ ψ)

6. $((p \wedge q) \vee \neg r)$ 5, 4, (◖ ∧ ψ)

Notice that we start with the smallest Boolean expression (namely, Boolean variables) and work our way up. Look at line number 4. The reason is ñ3, ¬◖ò. This means that we are using the rule ¬◖ to create line 4, where ◖ is from line 3. This is just the second part of the definition being applied. And we use lines 1 and 2 and rule (◖ ∧ ɣ) to create the expression in line 5. Again, this rule comes from part 3 of the definition.

Definition A construction of a Boolean expression is a list of steps, where each line is either a Boolean variable or it uses a connective (e.g., ¬, ∧, ∨, Ÿ or ź) to connect two other Boolean expressions (they may be the same), with line numbers that are less than itself. Each line is a valid Boolean expression.

For example, look at the construction above. If we have to add a 7th step to the construction, we would have two choices. Either we could introduce a Boolean variable (we could always do this) or we could use a connective and find a Boolean expression that is already on the list and add it. For example, we could place a ¬ in front of $\neg r$ (from step 4) and produce $\neg\neg r$.

The point of this exercise was to explain how to convince someone else that $((p \wedge q) \vee \neg r)$ is a Boolean expression. It is a kind of proof and uses the definition of Boolean expressions as reasons or justifications for each step.

The only difficulty with using this (and it is a small one) is that it is sometimes easier seeing an expression top-down than bottom-up. That is, it is intuitively simpler to take a complicated expression like $((p \wedge q) \vee \neg r)$ and try to break it down to its two parts, $(p \wedge q)$ and $\neg r$.

MEANING OF BOOLEAN EXPRESSIONS

One use of logic is as a means of deciding what things are true, given that certain facts are already true. Logic provides us a framework for deducing new things that are true. However, this deduction is based on form. For example, we might say that either $x > 0$ or x Ò 0 and also that x is not greater than 0. Given these two facts, we should be able to conclude that x Ò0.

Now both arguments actually have the same form that is, we basically said ◖ or ɣ is true and then ◖ is not true, therefore we concluded ɣ is true. In the first example, ◖ was $x > 0$ and ◖ was x Ò 0, while in the second example ◖ was ñthe capital of India is Mumbaiò and ɣ was ñthe capital of India is New Delhiò. In both examples, there was a similar form of the argument and the conclusion that we drew was purely based on the form.

This is actually at the heart of logic. (Some) English arguments can be translated into Boolean expressions, and then we can apply rules of logic to determine whether the arguments make sense, atleast, based on their form.

We know that Boolean variables are the building blocks of Boolean expressions. Boolean variables like p generally stand for either English or mathematical propositions.

Definition A proposition is something that is either true or false but not both.

Not all English sentences are propositions. For example, the sentence ñRun awayò cannot really be said to be true nor false. Not all mathematical ñsentencesò are propositions either. For example, $x > y$ is neither true nor false. We would need to know the values of x and y before we could draw the conclusion. Actually, we do not have to be this strict, $x > y$ is either true or false, so in some sense, we can consider it a proposition.

Once the translation has been made from English sentences or mathematical sentences into Boolean expressions, then we generally do not care what the original sentence means. We can make conclusions based on the Boolean expressions. We shall get into the details soon.

1.4.1 Conjunctions

The most basic Boolean expression is a Boolean variable, which is either true or false. Throughout this section, we shall refer to two propositions: p and q.

p I own a cat.

q I own a dog.

One way to make a more complicated sentence is to connect two sentences with ñandò. For example, ñI own a catò AND ñI own a dogò. In propositional calculus (which is what we are studying now), our purpose is to determine when expressions or sentences are true. So, when is the entire expression ñI own a cat AND I own a dogò true? Intuitively, we would say it is true if both parts are true.

Now letôs look at the Boolean expression equivalent of that same sentence. It happens to be $(p \wedge q)$ (again, notice the use of parentheses). The symbol for AND is $\wedge$, which we can pronounce as AND. If it helps us to remember, the $\wedge$ symbol looks sort of like an ñAò, which is the first letter of AND. Sometimes, mathematicians say that $(p \wedge q)$ is a *conjunction* of p and q and that p and q are conjuncts of the conjunction. Despite the fancy name, the work ñconjunctionò does come up often enough and hence we ought to remember it.

So, when is $(p \wedge q)$ true? When both p and q are true. If either is false, then the whole expression is false. We can actually summarize this in a *truth table*. A truth table tells us the ñtruthò of a Boolean expression given that we assign either true or false to each of the Boolean variables.

Given two different Boolean variables, p and q, there are four different ways to assign truth values to them. Each of the four ways is listed below. T stands for true, while F stands for false. If we look at the column for variable q, we will see that it alternates T, then F, then T, then F, whereas the column with p to its immediate left alternates, T, T, then F, F. If we had another variable, r and placed it to the left of p, it would alternate T, T, T, T then F, F, F, F.

There is a pattern. Starting from the rightmost Boolean variable, we will alternate every turn T, F, T, F. The next column to its left will alternate T, T, F, F, T, T, F, F, etc. The next one to its left will alternate T, T, T, T, F, F, F, F. As we move to the left, we repeat the Ts twice as many times as the previous column

and twice as may Fs. This pattern actually covers all possible ways of combining truth values for *n* Boolean variables.

Now, look at line 1 in the truth table. Look at the last column. We will notice that the entry has the value T, which means that when *p* is assigned T and *q* is assigned T, then ($p \wedge q$) is true as well. This is just a formalization of what we said before, ($p \wedge q$) is only true when *p* and *q* are both true (that is, both assigned to true).

The key point is to notice that we can find out the truth value of a complicated expression by knowing the truth value of the parts that make it up. This is really no different from arithmetic expressions. For example, if we had the expression (x + y) ī *z,* then we could tell that the value for this expression, provided if we knew the values for each of the variables. It is the same with Boolean expressions. If we know the truth values of the Boolean variables, then using truth tables, we can determine the truth value of a Boolean expression.

Now, we used *p* and *q* for the truth table above. However, there was nothing special about using those two Boolean variables. Any two different Boolean variables would have worked. In fact, any two Boolean *expressions* would have worked. We could have replaced *p* with ◖ and *q* with ɣ and ($p \wedge q$) with (◖ ∧ ɣ) and the truth table would still have been fine.

1.4.2 Disjunctions

Instead of saying ñI own a catò AND ñI own a dogò, we could connect the two statements with OR, as in, ñI own a catò OR ñI own a dogò. When would this statement be true? It would be true if I owned either a cat or a dog. That is, only one of the two statements has to be true. What happens if both are true? Then is the *entire* statement ñI own a cat OR I own a dogò true? Going by propositional logic, we will say yes. That is, the entire OR statement is true if one or the other statement or both are true.

We use the symbol ∨ to represent OR. So ($p \vee q$) (again, notice the parentheses) is the same as *p* OR *q* and the whole expression is true if *p* is true or *q* is true or both are true. It is false if both *p* and *q* are false. Sometimes the expression ($p \vee q$) is called a *disjunction* (with a ∧, it was a conjunction) and *p* and *q* are the disjuncts of the disjunction. Any Boolean expression (or sub expression) can be called a disjunction if it has the pattern (◖ ∨ ɣ).

The use of ñORò in propositional calculus actually contrasts with the way we normally use it in English. For example, if A said, ñI will go to the Cinema OR I will go to the Gardenò. Usually it means, I will go to one or the other, but NOT both. This kind of ñORò is called as *exclusive* OR, while the one we use in propositional calculus is called an *inclusive* OR. We will almost always use the inclusive OR (the exclusive OR can be defined using inclusive ORs and negations, which will be introduced in the next section).

Here is the truth table for OR.

Notice the rightmost column of this truth table and compare it to the rightmost column of the truth table of ($p \wedge q$). In the case of conjunction (i.e., AND), ($p \wedge q$) is true in only one case, namely, when *p*

and q are both true. It is false in all other cases. However, for ($p \vee q$) (read p OR q), it is true in all cases except when both p and q are false. In other words, only one of either p or q has to be true for the entire expression ($p \vee q$) to be true.

The main point covered so far:

The symbols we have seen: {∧, ∨, Ÿ , ź , ¬} are often called *connectives* because they connect two Boolean expressions (although in the case of ¬, itôs only attached to a single Boolean expression).

1.4.3 Negations

The symbol ¬, (pronounced ñnotò) is like a negative sign in arithmetic. So, if we have p (ñI own a catò), the ¬p (notice there are *no* parentheses) can be read as ñNot I own a catò, or ñIt is not the case that I own a catò (in which case the ¬ could be read as ñit is not the case thatò). Both of these sound awkward, but the idea is to use the original sentence and attach something before it, just like the connective. In English, it sounds more correct to say ñI do not own a catò.

Unlike ∧ and ∨, ¬ only attaches to a single Boolean expression. So, the truth table is actually smaller for ¬ since there is only one Boolean variable to worry about.

Line	p	¬p
1	T	F
2	F	T

This should be an easy truth table to understand. If p is true, then ¬p is false. The reverse holds as well. If p is false, then ¬p is true.

CONSTRUCTION OF TRUTH TABLES

Suppose we are given a Boolean expression, say, (($p \wedge q$) ∨ ¬p). We want to know whether this expression is true or false.

We introduce a function, v. This function will be called a *valuation*. If we were to write this function signature in pseudo-C++ code, it would look like

```
boolean v( boolExpr x );
```

In words, this function takes a Boolean expression as input (think of it as a class) and returns back a Boolean value, that is, it returns either true or false.

Let us formally define a valuation.

Definition A valuation (also called, a truth value function or a truth value assignment) is a function which assigns a truth value (that is, true or false) for a Boolean expression, under the following restrictions.

$v((\phi \wedge \psi))$	$= \min(v(\phi), v(\psi))$
$v((\phi \vee \psi))$	$= \max(v(\phi), v(\psi))$
$v(\neg\phi)$	$= 0$, if $v(\phi) = 1$
	$= 1$, if $v(\phi) = 0$
$v((\phi \ddot{Y} \psi))$	$= 1$, if $v(\phi) = F$ or $v(\psi) = T$
	$= 0$, otherwise
$v((\phi \hat{Y} \psi))$	$= 1$, if $v(\phi) = v(\psi)$
	$= 0$, otherwise

The interesting thing is that because of these restrictions, once a truth value function has been defined for all the Boolean variables in a Boolean expression, the truth value for the Boolean expression (and all its sub expressions) are defined as well.

Let us take a closer look at the definition. We shall take it line by line. In the first line, we have

$v((\phi \wedge \chi)) = \min(v(\phi), v(\chi))$

This says that if we want to find the truth value of $(\phi \wedge \chi)$, then we have to find the truth value of ϕ (that is, $v(\phi)$) and the truth value of χ (that is, $v(\chi)$). We take the ñminimumò of $v(\phi)$ and $v(\chi)$. How does one take the minimum of the two? If we treat false as the number 0 and true as the number 1, then taking the minimum of two numbers makes sense. But is it an accurate translation of AND?

Let us think about this for a moment. When is $(\phi \wedge \chi)$ true? When ϕ is true AND when χ is true. If we think of true as being the number 1, then we are asking what the minimum of 1 and 1 is. And the minimum of those two numbers is 1. If we translate it back, we get true. That seems to work.

Now, when is $(\phi \wedge \chi)$ false? When either ϕ or χ is false, that is, when $v(\chi) = F$ or (and this is an inclusive or) when $v(\chi) = F$. So, letôs think about this. If one of the two is false, then it has a value of 0. The minimum of 0 and anything else is 0. Why? Well, since truth values are either 0 (for false) or 1 (for true), we can only take the minimum of 0 and some other number. That number could be 0, in which case the minimum is 0, or it could be 1, in which case the minimum is still 0. So, the minimum of 0 and any other number (restricted to 0 or 1) is 0. And that makes sense too because we want $(v(\phi \wedge \chi))$ to be false (i.e., 0) when either $v(\phi)$ or $v(\chi)$ is false.

If we treat false as 0 and true as 1, then we can show

$v((\phi \vee \chi)) = \max(v(\phi), v(\chi))$

makes sense too. *max* is the function that takes the maximum of two numbers (in this case, we need to treat the truth values like numbers).

The real point of this is to show that to find out the truth value of a Boolean expression (i.e., to find out the value of $v(\phi)$, we need to find out the value of the smaller sub expressions.) For example, to find

$v(\neg \blacklozenge)$, we need to know the value of $v(\blacklozenge)$. And to find out $v(\blacklozenge)$ we need to see what pattern $\blacklozenge$ follows (is it a negation, conjunction, or disjunction?) and recursively apply the definition. At each step, we break down the equation into smaller and smaller sub expressions until we reach the smallest sub expression, which happens to be a Boolean variable.

To find the truth value of a Boolean expression, we just need to know the truth value of the Boolean variables in that expression.

Back to Derivations

Based on the insight of the previous section, we now return to our problem. We want to construct a truth table for $((p \wedge q) \vee \neg p)$. To do so, we need to find the Boolean variables in this expression. This is easy as there are only p and q.

This is how we will derive the Boolean expression. The reason will become clear, but we intend to use it to construct a truth table.

Here is the derivation.

Expression	*Reason*
1. p	Boolean variable
2. q	Boolean variable
3. $\neg p$	1, $\neg \blacklozenge$
4. $(p \wedge q)$	1, 2, $(\blacklozenge \wedge \psi)$
5. $((p \wedge q) \vee \neg p)$	4, 3, $(\blacklozenge \wedge \psi)$

We will create one column in the truth table for each line in the derivation. How many rows do we use? If n is the number of Boolean variables (in this case, 2), then 2^n is the number of rows (in this case, $2^2 = 4$ rows).

Now we just fill in the columns one after another. We know that column 3 is derived from column 1. So, we can just get column 3 by ñnegatingò the values in column 1. Compare the values of column 1 and column 3. We are basically applying the definition of $\vee$ from the last section.

Column 4 is based on columns 1 and 2. We get the values by adding them.

Finally, column 5 is constructed from columns 4 and 3, respectively.

And that is how it is done!

Valuations and Truth Tables

If we know the truth values for the variables, then the value of the Boolean expression is known as well. There is a simple analog to arithmetic expressions. In an expression like $(x + y)$ ī z, we know the value once we know what *value* each variable has been assigned. For example, if $x = 4$, $y = 3$, and $z = 5$, the expressionôs value is 2. If we change the values of the variables, the result of the expression changes too. However, the result is completely defined by the values assigned to the variables.

How does v relate to truth tables? Essentially each row of a truth table corresponds to a separate v. For example, consider the last truth table in the previous section. Row 1 defines $v(p) = T$ and $v(q) = T$. Given this, $v((p \wedge q) \vee \neg p)$ must have the value *true*. Row 2 defines a different v, one where $v(p) = T$, but $v(q) = F$, and where $v((p \wedge q) \vee \neg p)$ consequently has the value *false*.

How Many Rows

If there are n Boolean variables, then there will be 2^n rows in the truth table. Why?

We start with one Boolean variable, say this is *r*. *r* can either be true or false. So, a truth table with just one variable has two lines. One when the value is true and one when it is false.

Line	*r*
1	T
2	F

Now we add a second variable, *q*. *q* can also be either true or false. So, suppose *q* is true. Then, we get a truth table that looks like:

	1	2
Line	*q*	*r*
1	T	T
2	T	F

This still gives us two rows. Notice we have done nothing to the *r* column. *r* still has the two values, *T* and *F*. Meanwhile *q* has been set to true. However, what about the case when *q* is false? We should get two more rows for when *q* is false because *r* can still be true or false in that case.

So notice that rows 3 and 4 have F in column 1, (this is when *q* is false) and T in rows 1 and 2.

	1	2
Line	*q*	*r*
1	T	T
2	T	F
3	F	F
4	F	F

At this point, we have shown 4 different ways in which *q* and *r* can be assigned truth values. Now, letôs take it one step further. We add a third variable *p*. Suppose $p = T$ (or more accurately, $v(p) = T$). How many different combinations of truth values are there for *q* and *r*? Four, right? There were four combinations of truth values for *q* and *r* when we did not have *p*, so why should there be any more or less when $v(p) = T$? So, there should be 4 ways when *p* is true.

Now how many combinations of *q* and *r* are there when *p* is false? It is still 4, for the same reason. Now, we have a total of 8 rows. 4 rows when *p* is true (since there are 4 combinations of values for *q* and *r*) and 4 more when *p* is false. It will give a total of 8. Does this follow our formula? There are 3 variables, so there should be 2^3 rows and 2^3 equals 8, so it matches.

But the formula for the number of rows actually makes sense. Let us pretend we have *n* variables. This ought to create 2^n rows or equivalently, 2^n different ways to assign truth values to *n* variables. Now, we want to add one more variable. We shall call this new variable, *p*. That will make a total of $n + 1$ variables. We expect there to be 2^{n+1} rows.

We know there are 2^n different combinations for *n* rows. Now, there are still 2^n combinations when *p* is true. After all, why should *p* being true (in effect, it is a constant) affect the number of combinations of the other *n* variables? It should not and it does not. Then, there are 2^n combinations when *p* is false for the same reason. So there are 2^n combinations when *p* is true and 2^n combinations when *p* is false. Add the

two together and we get $2^n + 2^n = 2^{n+1}$. So, adding a new variable doubles the number of rows. We get one set of rows when the new variable is true and another set when it is false.

Making Truth Tables

Just because we know there are 2^n rows in a truth table, does not mean that it is easy to figure out a quick way to fill one up. So, here is the quick way to do it. We shall use three variables *p, q,* and *r*.

To start, fill in the right-most row with a Boolean variable. In this case, it is column 3 (the column with *r*). Start alternating T, F, T, F all the way down.

Go on to the column over to the left. This is column 2. Go down alternating T, T then F, F then repeat. So, 2 Ts, followed by two Fs and repeat.

Finally, move to column 1 and repeat 4 Ts, followed by 4 Fs. That is, T, T, T, T then F, F, F, F and repeat, if necessary.

In doing this we see a general pattern. Suppose we are in a particular column and are repeating the pattern of *n* Ts, followed by *n* Fs. If we move one column left, we double the number of Ts followed by double the number of Fs; that is, 2*n* Ts, followed by 2*n* Fs.

Why does this work? Well, it is related to the previous section. Think of T as being 1 and F as being 0. Look at row 1. There are three tôs in this row. Convert it to a binary number. This will be 111. Now, we go to row 2. Reading across, we read T, T, F. Converting to 1s and 0s, we get 110. In binary, this is 6. Now, as we progress down from row 1 to row 8, we will be counting backwards in binary, from 7 down to 0. So, essentially a truth table for *n* variables has one row for each *n* bit binary number. We shall learn about binary numbers later on, but this is the basic idea of how filling out a truth table works.

LOGICAL EQUIVALENCE

The idea of logical equivalence is deceptively simple. We are given two Boolean expressions. How do we determine if they are the same? The first question we need to ask is, ñWhat do we mean by ótheo?ò The two expressions are same if they generate the same truth table. We can be a little bit more formal. As we have said earlier, once we have defined the truth values of all the Boolean variables in a Boolean expression, we can know (or calculate) the truth value of the Boolean expression.

Now, suppose we had two different Boolean expressions, *a* (pronounced ñalphaò) and *b* (pronounced ñbetaò). Write down all the Boolean variables possible for both of them. Suppose *a* has *m* Boolean variables and *b* has *n* Boolean variables (usually, *m* = *n*, but this does not have to be the case). Now suppose that between the two, there are *k* unique variables. For example, *a* might have variables *p, q,* and

r, while b has variables, p and r_2. Between the two, there are 4 unique variables. In this example, k is 4. Thus, there are 2^k different ways of assigning truth values to these k unique variables.

For every function, v (there are 2^k of these, one for each row of the truth table), if $v(a) = v(b)$, then the a and b are logically equivalent.

Let us consider a simple example. We want to show that p is logically equivalent to $\neg\neg p$. In arithmetic, this is the equivalent of saying $x = ï ï x$. While this is intuitively obvious, it is better to have a method to determine when expressions are logically equivalent.

There is only one Boolean variable in both expressions, namely p. So, there are $2^1 = 2$ ways to assign truth values to p. We can either assign it true or false. Now, all we do is construct the truth table for both expressions. The result looks like:

Columns 1 and 3 are identical, which means the two are logically equivalent.

There is a stranger example. Many times, both expressions we wish to show as logically equivalent have the same Boolean variables. This makes sense, because if the variables were different, say, p as one Boolean expression and q as another, then we could find a function v, which sets $v(p) = T$ and $v(q) = F$. Hence, there would exist some function v such that $v(p)$ Í $v(q)$ and thus the two would not be logically equivalent.

However, sometimes, even though a Boolean expression contains a variable, it does not really depend on that variable. For example, consider $(p \wedge \neg p)$. Even if we do not know the value of p, we know $(p \vee \neg p)$ is true. Why? For $(p \vee \neg p)$ to be true, either p or $\neg p$ must be true. That is, if one is true, the other must be false. So, one of the two is always guaranteed to be true. In other words, it does not matter whether p is true or false, $(p \vee \neg p)$ is always true. This means that this expression does not really depend on p.

For a Boolean expression to depend on a variable, the truth value of the expression must change at some point, if p changes its value. That is, there must be some assignment of truth values to the other variables in an expression, a, such that $v(a)$ Í vNj$alpha$). In this case, v might represent the valuation where p is true and vNywould then represent the valuation where p is false. For all other variables, v and vNgive the same truth values.

So, let us show that $(p \vee \neg p)$ and $(q \vee \neg q)$ are logically equivalent. There are two unique variables between the two of them: p and q. Now, we construct the truth table for both expressions.

Notice that columns 5 and 6 are identical. When two columns are identical in this manner, they are logically equivalent.

To show two expressions are not logically equivalent, we just need a single valuation, v, where $v(a)$ does not equal $v(b)$. Or equivalently, we need to find a single line in a truth table, where one expression's truth value is T, while the other's is false.

Now we show that $(p \wedge q)$ and $(p \vee q)$ are not logically equivalent. Again, we write up the truth tables for both.

Just by inspection, we should be able to tell that columns 3 and 4 are not the same. For example, we can look at row 2 where $v(p) = T$ and $v(q) = F$, so the result is that $v((p \wedge q)) = F$, but $v((p \vee q)) = T$. Row 3 also gives a case where $v((p \wedge q))$ Í $v((p \vee q))$. The two expressions agree in rows 1 and 4. Thus, we conclude that $(p \wedge q)$ and $(p \vee q)$ are not logically equivalent.

TAUTOLOGIES AND CONTRADICTIONS

Definition A tautology is a Boolean expression which always results in a true result, regardless of what the Boolean variables in the expression are assigned to.

We saw this earlier on, with the expression $(p \vee \neg p)$. If $v(p) = T$, then the whole expression is true. If $v(p) = F$, then also the whole expression is true. Basically, a tautology means that the result of the expression is independent of whatever Boolean variables have been assigned the result is always true.

Definition A contradiction is a Boolean expression which always results in a false result, regardless of what the Boolean variables in the expression are assigned to.

A contradiction is just the opposite of a tautology, in fact, given any Boolean expression that is a tautology, we just have to negate it to get a contradiction. For example, $(p \vee \neg p)$ is a tautology. The negation of that, $\neg(p \vee \neg p)$, is a contradiction. We can apply De Morganôs law and get $(\neg p \wedge \neg\neg p)$ and by using the simplification for double negation result in $(\neg p \wedge p)$. So, since all of these are logically equivalent, then $(\neg p \wedge p)$ is also a contradiction.

CONTRADICTION RULE

$\neg p$ is true and then deduce a contradiction, then p is true. The idea runs something like this: one generally believes math is consistent that is, we do not derive contradictions using the rules of logic (the most common contradiction is to derive $\neg q$ when we also know that q happens to be true). So, when we try to prove p, we try to assume $\neg p$ and if this leads to a contradiction, then we know that $\neg p$ cannot be true, since our system avoids contradiction and thus if $\neg p$ is not true, it is false, and if it is false, then p must be true. This is often the line of reasoning used in a proof by contradiction.

UNDERSTANDING QUANTIFIERS

What does $\forall x\ P(x)$ mean? We can read $\forall$ as ñfor allò, so the entire statement can be read as ñfor all x, P of xò. If we add a domain, things can be made clearer. So, let, D, our domain, be the set {2, 4, 6, 8, 10}.

Then the statement $\forall x \quad D\ P\{x)$ makes more sense. We expect every single x picked from D to have

the property *P*. In fact, a very convenient way to think about ∀*x* *D* *P*(*x*) is to expand it out using ANDs.

For example, the expansion of ∀*x* *D* *P*(*x*) would be:

P(2) ∧ *P*(4) ∧ *P*(6) ∧ *P*(8) ∧ *P*(10).

Every *x* in *D* having property *P* means *P*(2) is true AND *P*(4) is true AND *P*(6) is true AND *P*(8) is true AND *P*(10) is true. In general, we will not be able to write it all out like this because *D* could be infinite, but it is a correct way to think about what ∀ really means.

What does the expansion for ∃*x* *D* *P*(*x*) mean? ∃ is read as ñthere existsò, so the entire statement reads as ñthere exists an *x* in *D, P* of *x* (or such that *P*(*x*)) holds true. The phrase ñThere existsò should give the hint that at least one *x* from *D* has the property *P*. And if the previous example for ∀ used ANDs,

what do we think ∃ uses? If we said ORs, that would be correct. Using the same set *D* as before, ∃*x* *D* *P*(*x*) can be expanded to:

P(2) ∨ *P*(4) ∨ *P*(6) ∨ *P*(8) ∨ *P*(10).

This disjunction is true when at least one of the *P*s is true and that should make sense based on our intuition of what ñthere existsò means (that is, it means at least one exists).

We will normally not be able to expand out universal quantifiers (∀) or existential quantifiers (∃) in this manner because *D*, the domain, may be infinite. Nevertheless, it is useful to think about an expansion when trying to get a feel of what quantifiers mean.

REFERENCE

1. **TEXT BOOKS:**
2. [T1]Rosen, K.H., Discrete Mathematics and its Applications, McGraw Hill, (2006) 6th ed.
3. [T2]Kolman, Busby and Ross, "Discrete Mathematical Structure", PHI, 1996.
4. [T3]Babu Ram, "Discrete Mathematics", Pearson Education

REFERENCE BOOKS:

5. [T1]S.K. Sarkar, "Discrete Maths"; S. Chand & Co., 2000.
6. [T2]Tremblay, J.P. and Manohar, R., Discrete Mathematical Structures with Applications to Computer Science, Tata McGraw Hill, (2007).
7. www.cse.ohio-state.edu/~gurari/theory-bk/theory-bk-appendixse1.html
8. www.regentsprep.org/Regents/math/algtrig/ATP5/L**function**.htm
9. www.seas.upenn.edu/~jean/cis160/cis260slides7.
10. www.**maths**isfun.com/data/**graphs**-index
11. www.seas.upenn.edu/~eeaton/teaching/cmsc471.../**PropositionalLogic**.pp...