

DISCRETE MATHEMATICS

MCA 104

SELF LEARNING MATERIAL



**DIRECTORATE
OF DISTANCE EDUCATION**

SWAMI VIVEKANAND SUBHARTI UNIVERSITY

MEERUT – 250 005,

UTTAR PRADESH (INDIA)

SLM Module Developed By :

Author:

Reviewed by :

Assessed by:

Study Material Assessment Committee, as per the SVSU ordinance No. VI (2)

Copyright © **Gayatri Sales**

DISCLAIMER

No part of this publication which is material protected by this copyright notice may be reproduced or transmitted or utilized or stored in any form or by any means now known or hereinafter invented, electronic, digital or mechanical, including photocopying, scanning, recording or by any information storage or retrieval system, without prior permission from the publisher.

Information contained in this book has been published by Directorate of Distance Education and has been obtained by its authors from sources believed to be reliable and are correct to the best of their knowledge. However, the publisher and its author shall in no event be liable for any errors, omissions or damages arising out of use of this information and specially disclaim and implied warranties or merchantability or fitness for any particular use.

Published by: Gayatri Sales

Typeset at: Micron Computers

Printed at: Gayatri Sales, Meerut.

DISCRETE MATHEMATICS (MCA-104)

Unit-I

Relation: Type and compositions of relations, Pictorial representation of relations, Closures of relations, Equivalence relations, Partial ordering relation.

Function: Types, Composition of function, Recursively defined function

Mathematical Induction: Peano's axioms, Mathematical Induction

Discrete Numeric Functions and Generating functions

Simple Recurrence relation with constant coefficients, Asymptotic Behavior of functions

Algebraic Structures: Properties, Semi group, Monoid Group, Abelian group, properties of group, Subgroup, Cyclic group, Cosets, Permutation groups, Homomorphism, Isomorphism and Automorphism of groups.

Unit –II

Propositional Logic: Proposition, First order logic, Basic logical operations, Tautologies, Contradictions, Algebra of Proposition, Logical implication, Logical equivalence, Normal forms, Inference Theory, Predicates and quantifiers, Posets, Hasse Diagram, **Lattices:** Introduction, Ordered set, Hasse diagram of partially ordered set, Consistent enumeration, Isomorphic ordered set, Well ordered set, Lattices, Properties of lattices, Bounded lattices, Distributive lattices, and Complemented lattices.

Unit-III

Introduction to defining language, Kleene Closure, Arithmetic expressions, Chomsky Hierarchy, Regular expressions, Generalized Transition graph.

Unit-IV

Conversion of regular expression to Finite Automata, NFA, DFA, Conversion of NFA to DFA, Optimizing DFA, FA with output: Moore machine, Mealy machine, Conversions.

Unit-V

Non-regular language: Pumping Lemma, Pushdown Automata, and Introduction to Turing Machine and its elementary applications to recognition of a language and computation of functions.

UNIT-I

Relation:

Type and compositions of relations

Let A , B , and C be sets, and let R be a relation from A to B and let S be a relation from B to C . That is, R is a subset of $A \times B$ and S is a subset of $B \times C$. Then R and S give rise to a relation from A to C indicated by $R \circ S$ and defined by:

$a (R \circ S)c$ if for some $b \in B$ we have aRb and bSc .

is,

$$R \circ S = \{(a, c) \mid \text{there exists } b \in B \text{ for which } (a, b) \in R \text{ and } (b, c) \in S\}$$

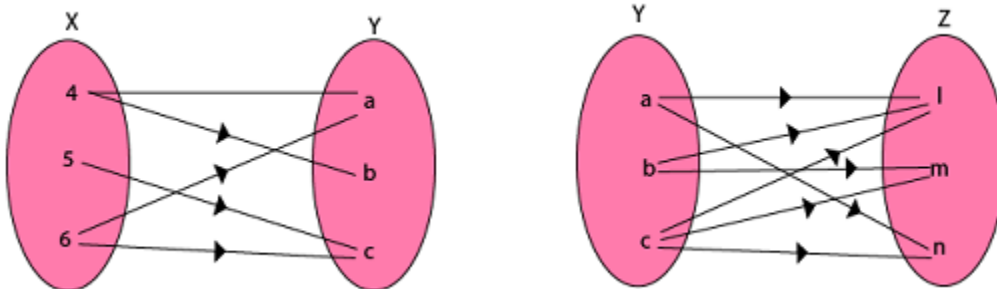
The relation $R \circ S$ is known the composition of R and S ; it is sometimes denoted simply by RS .

Let R is a relation on a set A , that is, R is a relation from a set A to itself. Then $R \circ R$, the composition of R with itself, is always represented. Also, $R \circ R$ is sometimes denoted by R^2 . Similarly, $R^3 = R^2 \circ R = R \circ R \circ R$, and so on. Thus R^n is defined for all positive n .

Example1: Let $X = \{4, 5, 6\}$, $Y = \{a, b, c\}$ and $Z = \{l, m, n\}$. Consider the relation R_1 from X to Y and R_2 from Y to Z .

$$R_1 = \{(4, a), (4, b), (5, c), (6, a), (6, c)\}$$

$$R_2 = \{(a, l), (a, n), (b, l), (b, m), (c, l), (c, m), (c, n)\}$$



Find the composition of relation (i) $R_1 \circ R_2$ (ii) $R_1 \circ R_1^{-1}$

Solution:

(i) The composition relation $R_1 \circ R_2$ as shown in fig:

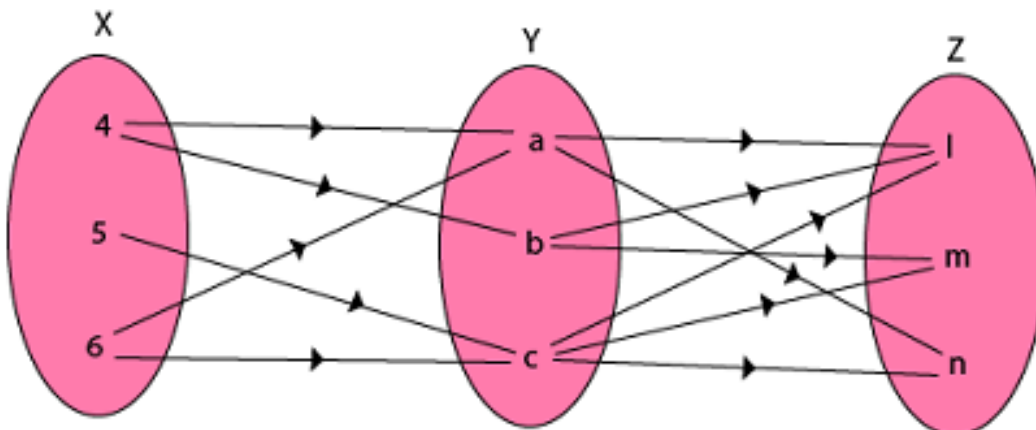


Fig : $R_1 \circ R_2$

$$R_1 \circ R_2 = \{(4, l), (4, n), (4, m), (5, l), (5, m), (5, n), (6, l), (6, m), (6, n)\}$$

(ii) The composition relation $R_1 \circ R_1^{-1}$ as shown in fig:

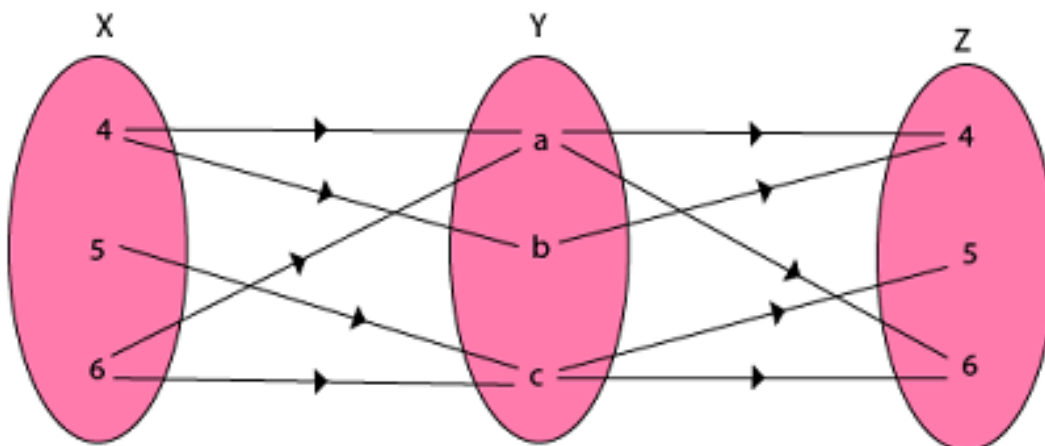


Fig : $R_1 \circ R_1^{-1}$

$$R_1 \circ R_1^{-1} = \{(4, 4), (5, 5), (5, 6), (6, 4), (6, 5), (4, 6), (6, 6)\}$$

Composition of Relations and Matrices

There is another way of finding $R \circ S$. Let M_R and M_S denote respectively the matrix representations of the relations R and S . Then

Example

Let $P = \{2, 3, 4, 5\}$. Consider the relation R and S on P defined by

$$R = \{(2, 2), (2, 3), (2, 4), (2, 5), (3, 4), (3, 5), (4, 5), (5, 3)\}$$

$$S = \{(2, 3), (2, 5), (3, 4), (3, 5), (4, 2), (4, 3), (4, 5), (5, 2), (5, 5)\}.$$

Find the matrices of the above relations.

Use matrices to find the following composition of the relation R and S .

- (i) $R \circ S$ (ii) $R \circ R$ (iii) $S \circ R$

Solution: The matrices of the relation R and S are shown in fig:

$$M_R = \begin{matrix} & \begin{matrix} 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left\{ \begin{matrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{matrix} \right\} \end{matrix} \quad \text{and} \quad M_S = \begin{matrix} & \begin{matrix} 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left\{ \begin{matrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{matrix} \right\} \end{matrix}$$

(i) To obtain the composition of relation R and S . First multiply M_R with M_S to obtain the matrix $M_R \times M_S$ as shown in fig:

The non zero entries in the matrix $M_R \times M_S$ tells the elements related in $R \circ S$. So,

$$M_R \times M_S = \begin{matrix} & \begin{matrix} 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left\{ \begin{matrix} 2 & 2 & 1 & 4 \\ 2 & 1 & 0 & 2 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{matrix} \right\} \end{matrix}$$

Hence the composition $R \circ S$ of the relation R and S is

1. $R \circ S = \{(2, 2), (2, 3), (2, 4), (3, 2), (3, 3), (4, 2), (4, 5), (5, 2), (5, 3), (5, 4), (5, 5)\}$.

(ii) First, multiply the matrix M_R by itself, as shown in fig

$$M_R \times M_R = \begin{matrix} & \begin{matrix} 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left\{ \begin{array}{cccc} 1 & 2 & 2 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array} \right\} \end{matrix}$$

Hence the composition $R \circ R$ of the relation R and S is

1. $R \circ R = \{(2, 2), (3, 2), (3, 3), (3, 4), (4, 2), (4, 5), (5, 2), (5, 3), (5, 5)\}$

(iii) Multiply the matrix M_S with M_R to obtain the matrix $M_S \times M_R$ as shown in fig:

$$M_S \times M_R = \begin{matrix} & \begin{matrix} 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \left\{ \begin{array}{cccc} 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{array} \right\} \end{matrix}$$

The non-zero entries in matrix $M_S \times M_R$ tells the elements related in $S \circ R$.

Hence the composition $S \circ R$ of the relation S and R is

1. $S \circ R = \{(2, 4), (2, 5), (3, 3), (3, 4), (3, 5), (4, 2), (4, 4), (4, 5), (5, 2), (5, 3), (5, 4), (5, 5)\}$.

Types of Relations

1. Reflexive Relation: A relation R on set A is said to be reflexive if $(a, a) \in R$ for every $a \in A$.

Example: If $A = \{1, 2, 3, 4\}$ then $R = \{(1, 1), (2, 2), (1, 3), (2, 4), (3, 3), (3, 4), (4, 4)\}$. Is a relation reflexive?

Solution: The relation is reflexive as for every $a \in A$, $(a, a) \in R$, i.e. $(1, 1), (2, 2), (3, 3), (4, 4) \in R$.

2. Irreflexive Relation: A relation R on set A is said to be irreflexive if $(a, a) \notin R$ for every $a \in A$.

Example: Let $A = \{1, 2, 3\}$ and $R = \{(1, 2), (2, 2), (3, 1), (1, 3)\}$. Is the relation R reflexive or irreflexive?

Solution: The relation R is not reflexive as for every $a \in A$, $(a, a) \notin R$, i.e., $(1, 1)$ and $(3, 3) \notin R$. The relation R is not irreflexive as $(a, a) \notin R$, for some $a \in A$, i.e., $(2, 2) \in R$.

3. Symmetric Relation: A relation R on set A is said to be symmetric iff $(a, b) \in R \Leftrightarrow (b, a) \in R$.

Example: Let $A = \{1, 2, 3\}$ and $R = \{(1, 1), (2, 2), (1, 2), (2, 1), (2, 3), (3, 2)\}$. Is a relation R symmetric or not?

Solution: The relation is symmetric as for every $(a, b) \in R$, we have $(b, a) \in R$, i.e., $(1, 2), (2, 1), (2, 3), (3, 2) \in R$ but not reflexive because $(3, 3) \notin R$.

Example of Symmetric Relation:

1. Relation \perp is symmetric since a line a is \perp to b , then b is \perp to a .
2. Also, Parallel is symmetric, since if a line a is \parallel to b then b is also \parallel to a .

Antisymmetric Relation: A relation R on a set A is antisymmetric iff $(a, b) \in R$ and $(b, a) \in R$ then $a = b$.

Example1: Let $A = \{1, 2, 3\}$ and $R = \{(1, 1), (2, 2)\}$. Is the relation R antisymmetric?

Solution: The relation R is antisymmetric as $a = b$ when (a, b) and (b, a) both belong to R .

Example2: Let $A = \{4, 5, 6\}$ and $R = \{(4, 4), (4, 5), (5, 4), (5, 6), (4, 6)\}$. Is the relation R antisymmetric?

Solution: The relation R is not antisymmetric as $4 \neq 5$ but $(4, 5)$ and $(5, 4)$ both belong to R .

5. Asymmetric Relation: A relation R on a set A is called an Asymmetric Relation if for every $(a, b) \in R$ implies that (b, a) does not belong to R .

6. Transitive Relations: A Relation R on set A is said to be transitive iff $(a, b) \in R$ and $(b, c) \in R \Leftrightarrow (a, c) \in R$.

Example1: Let $A = \{1, 2, 3\}$ and $R = \{(1, 2), (2, 1), (1, 1), (2, 2)\}$. Is the relation transitive?

Solution: The relation R is transitive as for every (a, b) (b, c) belong to R , we have $(a, c) \in R$ i.e, $(1, 2)$ $(2, 1) \in R \Rightarrow (1, 1) \in R$.

7. Identity Relation: Identity relation I on set A is reflexive, transitive and symmetric. So identity relation I is an Equivalence Relation.

Example: $A = \{1, 2, 3\} = \{(1, 1), (2, 2), (3, 3)\}$

8. Void Relation: It is given by $R: A \rightarrow B$ such that $R = \emptyset (\subseteq A \times B)$ is a null relation. Void Relation $R = \emptyset$ is symmetric and transitive but not reflexive.

9. Universal Relation: A relation $R: A \rightarrow B$ such that $R = A \times B (\subseteq A \times B)$ is a universal relation. Universal Relation from $A \rightarrow B$ is reflexive, symmetric and transitive. So this is an equivalence relation.

Pictorial representation of relations

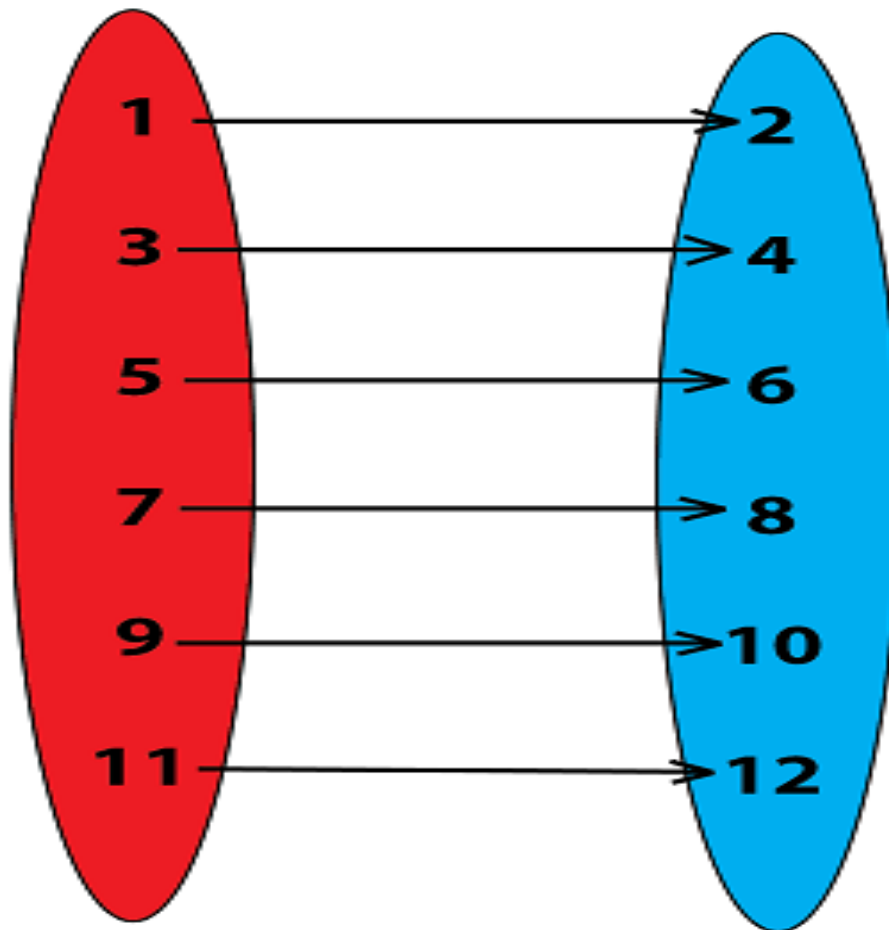
While you may be dealing with functions and relations, understanding them in pictorial forms makes it rather easier. Mapping diagrams help in the clearer understanding of the relations of numbers in one set of values and the other set of values.

Using Mapping Diagrams

Mapping diagrams prove to be useful while you're working with functions. They allow tracking the relationships between the inputs and the outputs. These diagrams can be used to make out which input values are tracked to which output values. They also help in ensuring that a function actually a function. For instance consider that we have the following set of pairs,

Inputs: {1, 3, 5, 7, 9, 11}

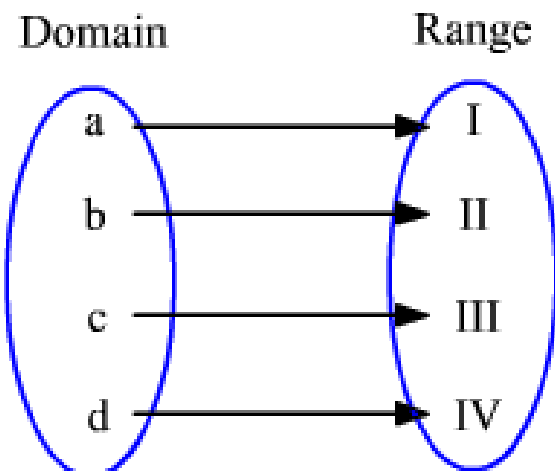
Outputs: {2, 4, 6, 8, 10, 12}



Here, the first number is to be considered as the input and the second number is considered as the output.

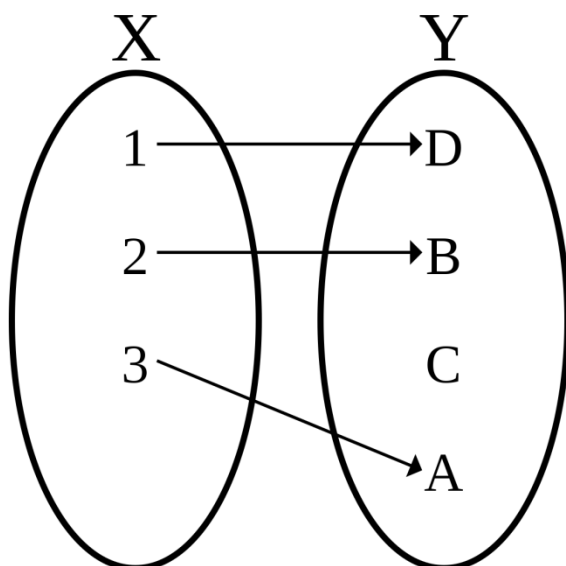
A function being a special relation where one element in the domain is paired exactly with only one element in the range set. Mapping shows the pattern in which the elements are paired. It is just like a flow chart for a specific function that displays the input and output for the same. Lines or arrows are drawn ascending from domain to range in order to represent the relation between two elements.

One-to-one Mapping



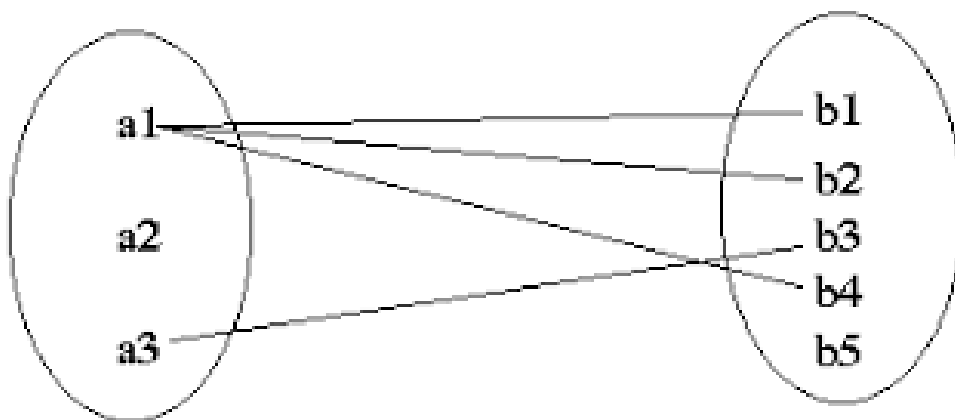
In the above image, the function represented by mapping above, here each element of the pair is ranged with exactly one element on the opposite sides. This is referred to as, one-to-one mapping.

Many-to-one Mapping



In the second image, the elements in the range set associate with probably more than one element in the range set. In case, the elements in range are known to be mapped with more than one element in the domain set, it would be called many-to-one mapping.

One-to-many Mapping



In this type of mapping, the first element in the domain set is known to be mapped with multiple elements in the range set. In a case like this when one element from the values of the domain set, the mapping indicates one-to-many relations. When one element has its relations with multiple elements, it cannot be termed as a function.

While keeping the elements scattered will make it complicated to understand relations and recognize whether or not they are functions, using pictorial representation like mapping will makes it rather sophisticated to take up the further steps with the mathematical procedures. Undeniably, the relation between various elements of the x values and y values.

Closures of relations

Consider a given set A, and the collection of all relations on A. Let P be a property of such relations, such as being symmetric or being transitive. A relation with property P will be called a P-relation. The P-closure of an arbitrary relation R on A, indicated P (R), is a P-relation such that

1. $R \subseteq P(R) \subseteq S$

(1) Reflexive and Symmetric Closures: The next theorem tells us how to obtain the reflexive and symmetric closures of a relation easily.

Theorem: Let R be a relation on a set A. Then:

- $R \cup \Delta_A$ is the reflexive closure of R
- $R \cup R^{-1}$ is the symmetric closure of R.

Example1:

1. Let $A = \{k, l, m\}$. Let R is a relation on A defined by
2. $R = \{(k, k), (k, l), (l, m), (m, k)\}$.

Find the reflexive closure of R.

Solution: $R \cup \Delta$ is the smallest relation having reflexive property, Hence,

$$R_F = R \cup \Delta = \{(k, k), (k, l), (l, l), (l, m), (m, m), (m, k)\}.$$

Example2: Consider the relation R on $A = \{4, 5, 6, 7\}$ defined by

1. $R = \{(4, 5), (5, 5), (5, 6), (6, 7), (7, 4), (7, 7)\}$

Find the symmetric closure of R.

Solution: The smallest relation containing R having the symmetric property is $R \cup R^{-1}$, i.e.

$$R_S = R \cup R^{-1} = \{(4, 5), (5, 4), (5, 5), (5, 6), (6, 5), (6, 7), (7, 6), (7, 4), (4, 7), (7, 7)\}.$$

(2)Transitive Closures: Consider a relation R on a set A. The transitive closure R of a relation R is the smallest transitive relation containing R.

Recall that $R^2 = R \circ R$ and $R^n = R^{n-1} \circ R$. We define

$$R^* = \bigcup_{i=1}^{\infty} R^i$$

The following Theorem applies:

Theorem1: R^* is the transitive closure of R

Suppose A is a finite set with n elements.

$$R^* = R \cup R^2 \cup \dots \cup R^n$$

Theorem 2: Let R be a relation on a set A with n elements. Then

$$\text{Transitive (R)} = R \cup R^2 \cup \dots \cup R^n$$

Example1: Consider the relation $R = \{(1, 2), (2, 3), (3, 3)\}$ on $A = \{1, 2, 3\}$. Then $R^2 = R \circ R = \{(1, 3), (2, 3), (3, 3)\}$ and $R^3 = R^2 \circ R = \{(1, 3), (2, 3), (3, 3)\}$ Accordingly, $\text{Transitive (R)} = \{(1, 2), (2, 3), (3, 3), (1, 3)\}$

Example2: Let $A = \{4, 6, 8, 10\}$ and $R = \{(4, 4), (4, 10), (6, 6), (6, 8), (8, 10)\}$ is a relation on set A. Determine transitive closure of R.

Solution: The matrix of relation R is shown in fig:

$$M_R = \begin{matrix} & \begin{matrix} 4 & 6 & 8 & 10 \end{matrix} \\ \begin{matrix} 4 \\ 6 \\ 8 \\ 10 \end{matrix} & \left\{ \begin{array}{cccc} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array} \right\} \end{matrix}$$

Now, find the powers of M_R as in fig:

$$M_{R^2} = \begin{matrix} & 4 & 6 & 8 & 10 \\ 4 & \left(\begin{array}{cccc} 1 & 0 & 0 & 1 \end{array} \right) \\ 6 & \left(\begin{array}{cccc} 0 & 1 & 1 & 1 \end{array} \right) \\ 8 & \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \right) \\ 10 & \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \right) \end{matrix} \quad M_{R^3} = \begin{matrix} & 4 & 6 & 8 & 10 \\ 4 & \left(\begin{array}{cccc} 1 & 0 & 0 & 1 \end{array} \right) \\ 6 & \left(\begin{array}{cccc} 0 & 1 & 1 & 1 \end{array} \right) \\ 8 & \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \right) \\ 10 & \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \right) \end{matrix}$$

$$M_{R^4} = \begin{matrix} & 4 & 6 & 8 & 10 \\ 4 & \left(\begin{array}{cccc} 1 & 0 & 0 & 1 \end{array} \right) \\ 6 & \left(\begin{array}{cccc} 0 & 1 & 1 & 1 \end{array} \right) \\ 8 & \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \right) \\ 10 & \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \right) \end{matrix}$$

Hence, the transitive closure of M_R is M_{R^*} as shown in Fig (where M_{R^*} is the ORing of a power of M_R).

$$M_{R^*} = M_R \vee M_{R^2} \vee M_{R^3} \vee M_{R^4}; \quad M_{R^*} = \begin{matrix} & 4 & 6 & 8 & 10 \\ 4 & \left(\begin{array}{cccc} 1 & 0 & 0 & 1 \end{array} \right) \\ 6 & \left(\begin{array}{cccc} 0 & 1 & 1 & 1 \end{array} \right) \\ 8 & \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \right) \\ 10 & \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \end{array} \right) \end{matrix}$$

Thus, $R^* = \{(4, 4), (4, 10), (6, 8), (6, 6), (6, 10), (8, 10)\}$.

Equivalence relations

In mathematics, relations and functions are the most important concepts. In class 11 and class 12, we have studied the important ideas which are covered in the relations and function. The concepts are used to solve the problems in different chapters like probability, differentiation, integration, and so on. In this article, let us discuss one of the concepts called “**Equivalence Relation**” with its definition, proofs, different properties along with the solved examples.

Table of Contents:

- Definition
- Proof
- Reflexive Property
- Symmetric Property
- Transitive Property
- Examples
- Practice Problems

Equivalence Relation Definition

A relation R on a set A is said to be an **equivalence relation** if and only if the relation R is reflexive, symmetric and transitive.

Reflexive: A relation is said to be reflexive, if $(a, a) \in R$, for every $a \in A$.

Symmetric: A relation is said to be symmetric, if $(a, b) \in R$, then $(b, a) \in R$.

Transitive: A relation is said to be transitive if $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$.

Equivalence relations can be explained in terms of the following examples:

- The sign of ‘is equal to’ on a set of numbers; for example, $1/3$ is equal to $3/9$.
- For a given set of triangles, the relation of ‘is similar to’ and ‘is congruent to’.
- For a given set of integers, the relation of ‘is congruent to, modulo n ’ shows equivalence.
- The image and domain are the same under a function, shows the relation of equivalence.
- For a set of all angles, ‘has the same cosine’.
- For a set of all real numbers, ‘has the same absolute value’.

- Sets
- Relations And Its Types
- Sets For Class 11
- Important Questions Class 11 Maths Chapter 1 Sets

Equivalence Relation Proof

Here is an equivalence relation example to prove the properties.

Let us assume that R be a relation on the set of ordered pairs of positive integers such that $((a, b), (c, d)) \in R$ if and only if $ad=bc$. Is R an equivalence relation?

In order to prove that R is an equivalence relation, we must show that R is reflexive, symmetric and transitive.

The Proof for the given condition is given below:

Reflexive Property

According to the reflexive property, if $(a, a) \in R$, for every $a \in A$

For all pairs of positive integers,

$((a, b), (a, b)) \in R$.

Clearly, we can say

$ab = ab$ for all positive integers.

Hence, the reflexive property is proved.

Symmetric Property

From the symmetric property,

if $(a, b) \in R$, then we can say $(b, a) \in R$

For the given condition,

if $((a, b), (c, d)) \in R$, then $((c, d), (a, b)) \in R$.

If $((a, b), (c, d)) \in R$, then $ad = bc$ and $cb = da$

since multiplication is commutative.

Therefore $((c, d), (a, b)) \in R$

Hence symmetric property is proved.

Transitive Property

From the transitive property,

if $(a, b) \in R$ and $(b, c) \in R$, then (a, c) also belongs to R

For the given set of ordered pairs of positive integers,

$((a, b), (c, d)) \in R$ and $((c, d), (e, f)) \in R$,

then $((a, b), (e, f)) \in R$.

Now, assume that $((a, b), (c, d)) \in R$ and $((c, d), (e, f)) \in R$.

Then we get, $ad = cb$ and $cf = de$.

The above relation implies that $a/b = c/d$ and that $c/d = e/f$,

so $a/b = e/f$ we get $af = be$.

Therefore $((a, b), (e, f)) \in R$.

Hence transitive property is proved.

Equivalence Relation Examples

Go through the equivalence relation examples and solutions provided here

Question 1:

Let assume that F is a relation on the set \mathbf{R} real numbers defined by xFy if and only if $x - y$ is an integer. Prove that F is an equivalence relation on \mathbf{R} .

Solution:

Reflexive: Consider x belongs to \mathbf{R} , then $x - x = 0$ which is an integer. Therefore xFx .

Symmetric: Consider x and y belongs to \mathbf{R} and xFy . Then $x - y$ is an integer. Thus, $y - x = -(x - y)$, $y - x$ is also an integer. Therefore yFx .

Transitive: Consider x and y belongs to \mathbf{R} , xFy and yFz . Therefore $x - y$ and $y - z$ are integers. According to the transitive property, $(x - y) + (y - z) = x - z$ is also an integer. So that xFz .

Thus, R is an equivalence relation on \mathbf{R} .

Question 2:

Show that the relation R is an equivalence relation in the set $A = \{ 1, 2, 3, 4, 5 \}$ given by the relation $R = \{ (a, b): |a - b| \text{ is even} \}$.

Solutio :

$R = \{ (a, b) : |a-b| \text{ is even} \}$. Where a, b belongs to A

Reflexive Property :

From the given relation,

$$|a - a| = |0| = 0$$

And 0 is always even.

Thus, $|a-a|$ is even

Therefore, (a, a) belongs to R

Hence R is Reflexive

Symmetric Property :

From the given relation,

$$|a - b| = |b - a|$$

We know that $|a - b| = |-(b - a)| = |b - a|$

Hence $|a - b|$ is even,

Then $|b - a|$ is also even.

Therefore, if $(a, b) \in R$, then (b, a) belongs to R

Hence R is symmetric.

Transitive Property :

If $|a-b|$ is even, then $(a-b)$ is even.

Similarly, if $|b-c|$ is even, then $(b-c)$ is also even.

Sum of even number is also even

So, we can write it as $a-b + b-c$ is even

Then, $a - c$ is also even

So,

$|a - b|$ and $|b - c|$ is even , then $|a-c|$ is even.

Therefore, if $(a, b) \in R$ and $(b, c) \in R$, then (a, c) also belongs to R

Hence R is transitive.

Practice problems on Equivalence Relation

Solve the practise problems on the equivalence relation given below:

1. Prove that the relation R is an equivalence relation, given that the set of complex numbers is defined by $z_1 R z_2 \Leftrightarrow [(z_1 - z_2)/(z_1 + z_2)]$ is real.
2. Show that the given relation R is an equivalence relation, which is defined by $(p, q) R (r, s) \Rightarrow (p+s)=(q+r)$
3. Check the reflexive, symmetric and transitive property of the relation $x R y$, if and only if y is divisible by x , where $x, y \in \mathbb{N}$.

Frequently Asked Questions on Equivalence Relation

What is meant by equivalence relation?

In mathematics, the relation R on the set A is said to be an equivalence relation, if the relation satisfies the properties, such as reflexive property, transitive property, and symmetric property.

What are the three different properties of the equivalence relation?

The three different properties of equivalence relation are:

Reflexive Property

Symmetric Property

Transitive Property

Explain reflexive, transitive and symmetric property.

A relation R is said to be reflexive, if $(x,x) \in R$, for every $x \in$ set A

A relation R is said to be symmetric, if $(x,y) \in R$, then $(y, x) \in R$

A relation R is said to be transitive, if $(x, y) \in R$ and $(y,z) \in R$, then $(x, z) \in R$

Can we say the empty relation is an equivalence relation?

We can say that the empty relation on the empty set is considered as an equivalence relation. But, the empty relation on the non-empty set is not considered as an equivalence relation.

Can we say every relation is a function?

No, every relation is not considered as a function, but every function is considered as a relation.

To learn equivalence relation easily and engagingly, register with BYJU'S – The Learning App and also watch interactive videos to get information for other Maths-related concepts.

Partial ordering relation

A relation R on a set A is called a partial order relation if it satisfies the following three properties:

1. Relation R is Reflexive, i.e. $aRa \forall a \in A$.
2. Relation R is Antisymmetric, i.e., aRb and $bRa \Rightarrow a = b$.
3. Relation R is transitive, i.e., aRb and $bRc \Rightarrow aRc$.

Example1: Show whether the relation $(x, y) \in R$, if, $x \geq y$ defined on the set of +ve integers is a partial order relation.

Solution: Consider the set $A = \{1, 2, 3, 4\}$ containing four +ve integers. Find the relation for this set such as $R = \{(2, 1), (3, 1), (3, 2), (4, 1), (4, 2), (4, 3), (1, 1), (2, 2), (3, 3), (4, 4)\}$.

Reflexive: The relation is reflexive as for every $a \in A$. $(a, a) \in R$, i.e. $(1, 1), (2, 2), (3, 3), (4, 4) \in R$.

Antisymmetric: The relation is antisymmetric as whenever (a, b) and $(b, a) \in R$, we have $a = b$.

Transitive: The relation is transitive as whenever (a, b) and $(b, c) \in R$, we have $(a, c) \in R$.

Example: $(4, 2) \in R$ and $(2, 1) \in R$, implies $(4, 1) \in R$.

As the relation is reflexive, antisymmetric and transitive. Hence, it is a partial order relation.

Example2: Show that the relation 'Divides' defined on \mathbb{N} is a partial order relation.

Solution:

Reflexive: We have a divides a , $\forall a \in \mathbb{N}$. Therefore, relation 'Divides' is reflexive.

Antisymmetric: Let $a, b, c \in \mathbb{N}$, such that a divides b . It implies b divides a iff $a = b$. So, the relation is antisymmetric.

Transitive: Let $a, b, c \in \mathbb{N}$, such that a divides b and b divides c .

Then a divides c . Hence the relation is transitive. Thus, the relation being reflexive, antisymmetric and transitive, the relation 'divides' is a partial order relation.

Example3: (a) The relation \subseteq of a set of inclusion is a partial ordering or any collection of sets since set inclusion has three desired properties:

1. $A \subseteq A$ for any set A .
2. If $A \subseteq B$ and $B \subseteq A$ then $B = A$.
3. If $A \subseteq B$ and $B \subseteq C$ then $A \subseteq C$

(b) The relation \leq on the set R of real no that is Reflexive, Antisymmetric and transitive.

(c) Relation \leq is a Partial Order Relation.

n-Ary Relations

By an n -ary relation, we mean a set of ordered n -tuples. For any set S , a subset of the product set S^n is called an n -ary relation on S . In particular, a subset of S^3 is called a ternary relation on S .

Partial Order Set (POSET):

The set A together with a partial order relation R on the set A and is denoted by (A, R) is called a partial orders set or POSET.

Total Order Relation

Consider the relation R on the set A . If it is also called the case that for all, $a, b \in A$, we have either $(a, b) \in R$ or $(b, a) \in R$ or $a = b$, then the relation R is known total order relation on set A .

Example: Show that the relation ' $<$ ' (less than) defined on N , the set of +ve integers is neither an equivalence relation nor partially ordered relation but is a total order relation.

Solution:

Reflexive: Let $a \in N$, then $a < a$
 \Rightarrow ' $<$ ' is not reflexive.

As, the relation ' $<$ ' (less than) is not reflexive, it is neither an equivalence relation nor the partial order relation.

But, as $\forall a, b \in N$, we have either $a < b$ or $b < a$ or $a = b$. So, the relation is a total order relation.

Equivalence Class

Consider, an equivalence relation R on a set A . The equivalence class of an element $a \in A$, is the set of elements of A to which element a is related. It is denoted by $[a]$.

Example: Let R be an equivalence relations on the set $A = \{4, 5, 6, 7\}$ defined by $R = \{(4, 4), (5, 5), (6, 6), (7, 7), (4, 6), (6, 4)\}$.

Determine its equivalence classes.

Solution: The equivalence classes are as follows:

$$\{4\} = \{6\} = \{4, 6\}$$

$$\{5\} = \{5\}$$

$$\{7\} = \{7\}.$$

Circular Relation

Consider a binary relation R on a set A . Relation R is called circular if $(a, b) \in R$ and $(b, c) \in R$ implies $(c, a) \in R$.

Example: Consider R is an equivalence relation. Show that R is reflexive and circular.

Solution: Reflexive: As, the relation, R is an equivalence relation. So, reflexivity is the property of an equivalence relation. Hence, R is reflexive.

Circular: Let $(a, b) \in R$ and $(b, c) \in R$

$$\Rightarrow (a, c) \in R \quad (\because R \text{ is transitive})$$

$$\Rightarrow (c, a) \in R \quad (\because R \text{ is symmetric})$$

Thus, R is Circular.

Compatible Relation

A binary relation R on a set A that is Reflexive and symmetric is called Compatible Relation.

Every Equivalence Relation is compatible, but every compatible relation need not be an equivalence.

Example: Set of a friend is compatible but may not be an equivalence relation.

Friend Friend

$a \rightarrow b,$ $b \rightarrow c$ but possible that a and c are not friends.

Function:

Types

Function, in mathematics, an expression, rule, or law that defines a relationship between one variable (the independent variable) and another variable (the dependent variable). Functions are ubiquitous in mathematics and are essential for formulating physical relationships in the sciences. The modern definition of function was first given in 1837 by the German mathematician Peter Dirichlet:

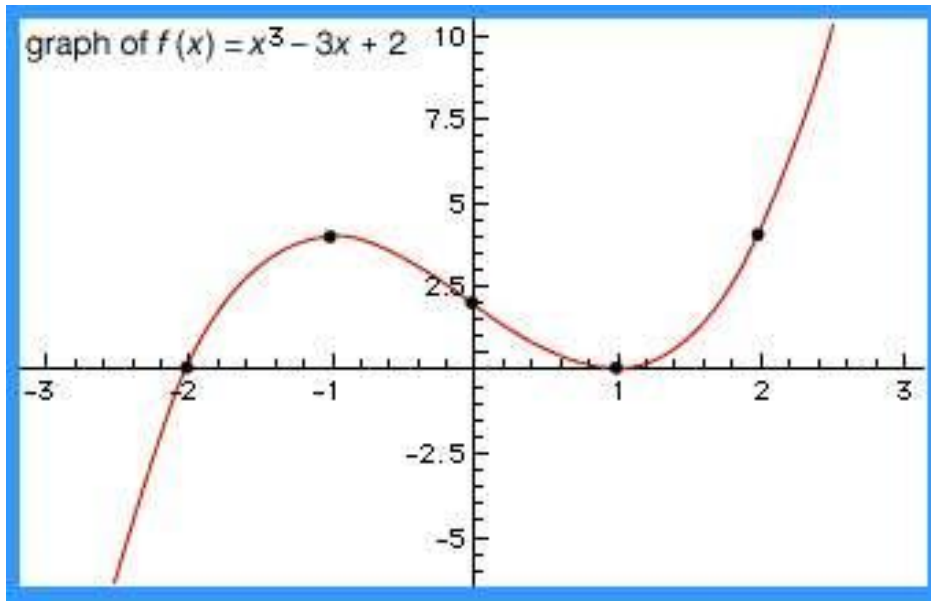
This relationship is commonly symbolized as $y = f(x)$. In addition to $f(x)$, other abbreviated symbols such as $g(x)$ and $P(x)$ are often used to represent functions of the independent variable x , especially when the nature of the function is unknown or unspecified.

Common Functions

Many widely used mathematical formulas are expressions of known functions. For example, the formula for the area of a circle, $A = \pi r^2$, gives the dependent variable A (the area) as a function of the independent variable r (the radius). Functions involving more than two variables also are common in mathematics, as can be seen in the formula for the area of a triangle, $A = bh/2$, which defines A as a function of both b (base) and h (height). In these examples, physical constraints force the independent variables to be positive numbers. When the independent variables are also allowed to take on negative values—thus, any real number—the functions are known as real-valued functions.

The formula for the area of a circle is an example of a polynomial function. The general form for such functions is $P(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$, where the coefficients ($a_0, a_1, a_2, \dots, a_n$) are given, x can be any real number, and all the powers of x are counting numbers (1, 2, 3, ...). (When the powers of x can be any real number, the result is known as an algebraic function.) Polynomial functions have been studied since the earliest times because of their versatility—practically any relationship involving real numbers can be closely approximated by a polynomial function. Polynomial functions are characterized by the highest power of the independent variable. Special names are commonly used for such powers from one to five—linear, quadratic, cubic, quartic, and quintic.

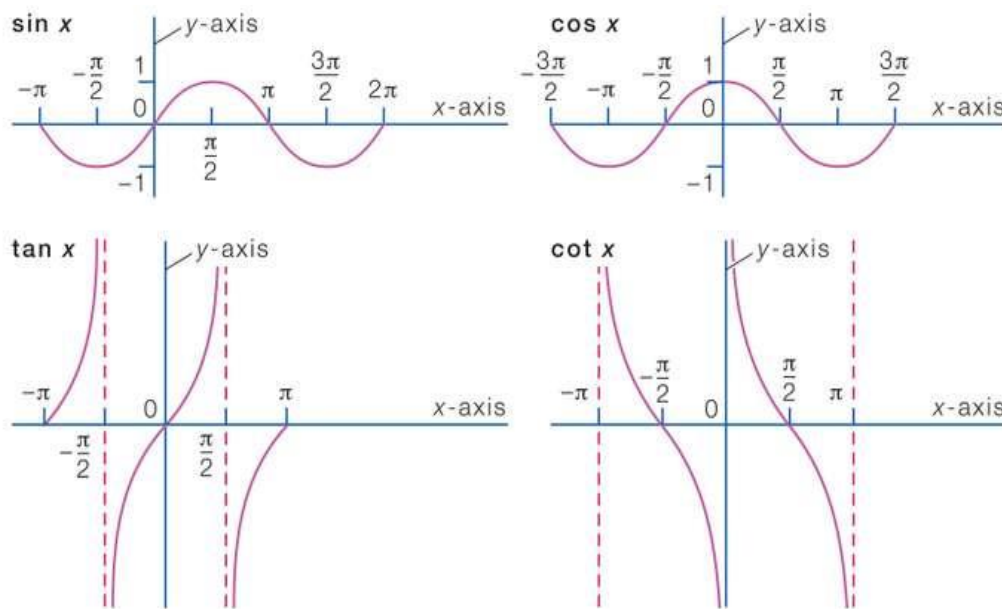
Polynomial functions may be given geometric representation by means of analytic geometry. The independent variable x is plotted along the x -axis (a horizontal line), and the dependent variable y is plotted along the y -axis (a vertical line). The graph of the function then consists of the points with coordinates (x, y) where $y = f(x)$. For example, the graph of the cubic equation $f(x) = x^3 - 3x + 2$ is shown in the figure.



Plot of the cubic equation $f(x) = x^3 - 3x + 2$. The plotted points are where changes in curvature occur.

Encyclopædia Britannica, Inc.

Another common type of function that has been studied since antiquity is the trigonometric functions, such as $\sin x$ and $\cos x$, where x is the measure of an angle (see figure). Because of their periodic nature, trigonometric functions are often used to model behaviour that repeats, or “cycles.” Nonalgebraic functions, such as exponential and trigonometric functions, are also known as transcendental functions.



© 2011 Encyclopædia Britannica, Inc.

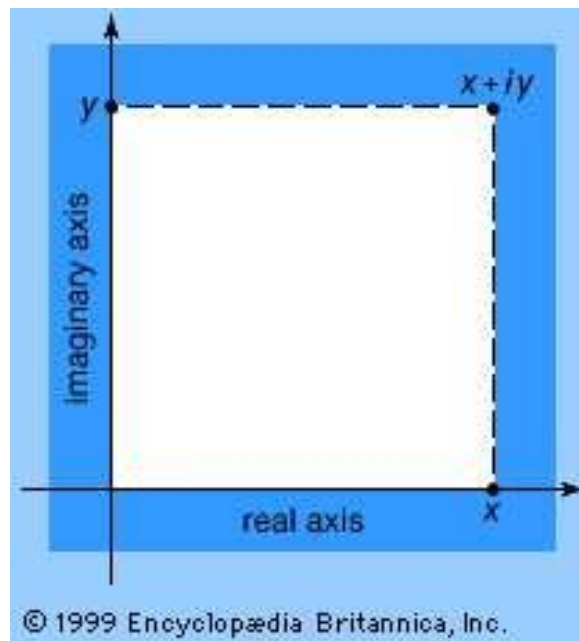
graphs of some trigonometric functions

Note that each of these functions is periodic. Thus, the sine and cosine functions repeat every 2π , and the tangent and cotangent functions repeat every π .

Encyclopædia Britannica, Inc.

Complex Functions

Practical applications of functions whose variables are complex numbers are not so easy to illustrate, but they are nevertheless very extensive. They occur, for example, in electrical engineering and aerodynamics. If the complex variable is represented in the form $z = x + iy$, where i is the imaginary unit (the square root of -1) and x and y are real variables (see figure), it is possible to split the complex function into real and imaginary parts: $f(z) = P(x, y) + iQ(x, y)$.



point in the complex plane

A point in the complex plane. Unlike real numbers, which can be located by a single signed (positive or negative) number along a number line, complex numbers require a plane with two axes, one axis for the real number component and one axis for the imaginary component. Although the complex plane looks like the ordinary two-dimensional plane, where each point is determined by an ordered pair of real numbers (x, y) , the point $x + iy$ is a single number.

Encyclopædia Britannica, Inc.

Inverse Functions

By interchanging the roles of the independent and dependent variables in a given function, one can obtain an inverse function. Inverse functions do what their name implies: they undo the action of a function to return a variable to its original state. Thus, if for a given function $f(x)$ there exists a function $g(y)$ such that $g(f(x)) = x$ and $f(g(y)) = y$, then g is called the inverse function of f and given the notation f^{-1} , where by convention the variables are interchanged. For example, the function $f(x) = 2x$ has the inverse function $f^{-1}(x) = x/2$.

Other Functional Expressions

A function may be defined by means of a power series. For example, the infinite series

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + \cdots$$
$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} \cdots$$
$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} \cdots$$

could be used to define these functions for all complex values of x . Other types of series and also infinite products may be used when convenient. An important case is the Fourier series, expressing a function in terms of

$$f(x) = a_0 + a_1 \cos x + a_2 \cos 2x + \cdots + b_1 \sin x + b_2 \sin 2x + \cdots$$

sines and cosines:

Such representations are of great importance in physics, particularly in the study of wave motion and other oscillatory phenomena.

Sometimes functions are most conveniently defined by means of differential equations. For example, $y = \sin x$ is the solution of the differential equation $d^2y/dx^2 + y = 0$ having $y = 0$, $dy/dx = 1$ when $x = 0$; $y = \cos x$ is the solution of the same equation having $y = 1$, $dy/dx = 0$ when $x = 0$.

Composition of function

In this lesson, I will go over eight (8) worked examples to illustrate the process involved in function composition.

If we are given two functions, it is possible to create or generate a “new” function by composing one into the other. The step involved is similar when a function is being evaluated for a given value. For instance, evaluate the function below for $x = 3x=3$.

$$f(x) = 4x^2 - 2x + 5$$

It is obvious that I need to replace each x by the given value then simplify.

$$f(x) = 4x^2 - 2x + 5$$

$$f(-3) = 4(-3)^2 - 2(-3) + 5$$

$$= 4(9) + 6 + 5$$

$$= 36 + 6 + 5$$

$$f(-3) = 47$$

General Rule of Composition of Function

Suppose the two given functions are f and g , the composition of $f \circ g$ is defined by

input
↓
 $f \circ g = f[g(x)]$

Also, the composition of $g \circ f$ is defined by

input
↓

$$g \circ f = g[f(x)]$$

Few notes about the symbolic “formula” above:

- The order in function composition matters! You always compose functions **from right to left**. Therefore, given a function, its input is always the one to its right side. In other words, the right function goes inside the left function.

LEFT

RIGHT

Function A ← **Function B**

outer function

inner function

- Notice in $f \circ g = f(\{g(x)\})$, the input or “inner function” is function g because it is to the right of function f which is the main or “outer function”.
- In terms of the order of composition, do you see the same pattern in $g \circ f = g(\{f(x)\})$? That’s right! The function f is the inner function of the outer function g .

Let us go over a few examples to see how function composition works. You will realize later that it is simply an exercise of algebraic substitution and simplification.

Examples of How to Compose Functions

Example 1: Perform the indicated function composition:

$$\left. \begin{aligned} f(x) &= 2x^2 - 3x - 1 \\ g(x) &= -x + 5 \end{aligned} \right\} \text{find } f \circ g$$

The order of composition is important. Notice that in $f \circ g$, we want the function $g(x)$ to be the input of the main function $f(x)$. It should look like this:

$$f \circ g = f[g(x)]$$

I start by writing down the main or outer function $f(x)$, and in every instance of x , I will substitute the full value of $g(x)$. Then, I’ll do whatever is needed to simplify the expressions such as squaring the binomial, applying the distributive property, and combining like terms. Other than that, there’s really nothing much to it.

Let me show you what I meant by that.

Example 2: Perform the indicated function composition:

$$\left. \begin{aligned} f(x) &= -10x^2 + 6x - 4 \\ g(x) &= \frac{1}{2}x + 2 \end{aligned} \right\} \text{find } g \circ f$$

I need to find the composite function $g \circ f$ which means function f is the input of function g .

Recursively defined function

In mathematics and computer science, a **recursive definition**, or **inductive definition**, is used to define the elements in a set in terms of other elements in the set (Aczel 1977:740ff). Some examples of recursively-definable objects include factorials, natural numbers, Fibonacci numbers, and the Cantor ternary set.^[1]

A recursive definition of a function defines values of the function for some inputs in terms of the values of the same function for other (usually smaller) inputs. For example, the factorial function $n!$ is defined by the rules

$$\begin{aligned} 0! &= 1. \\ (n + 1)! &= (n + 1) \cdot n!. \end{aligned}$$

This definition is valid for each natural number n , because the recursion eventually reaches the **base case** of 0. The definition may also be thought of as giving a procedure for computing the value of the function $n!$, starting from $n = 0$ and proceeding onwards with $n = 1$, $n = 2$, $n = 3$ etc.

The recursion theorem states that such a definition indeed defines a function that is unique. The proof uses mathematical induction.^[2]

An inductive definition of a set describes the elements in a set in terms of other elements in the set. For example, one definition of the set \mathbf{N} of natural numbers is:

1. 1 is in \mathbf{N} .
2. If an element n is in \mathbf{N} then $n + 1$ is in \mathbf{N} .
3. \mathbf{N} is the intersection of all sets satisfying (1) and (2).

There are many sets that satisfy (1) and (2) – for example, the set $\{1, 1.649, 2, 2.649, 3, 3.649, \dots\}$ satisfies the definition. However, condition (3) specifies the set of natural numbers by removing the sets with extraneous members. Note that this definition assumes that \mathbf{N} is contained in a larger set (such as the set of real numbers) — in which the operation $+$ is defined.

Properties of recursively defined functions and sets can often be proved by an induction principle that follows the recursive definition. For example, the definition of the natural numbers presented here directly implies the principle of mathematical induction for natural numbers: if a property holds of the natural number 0 (or 1), and the property holds of $n+1$ whenever it holds of n , then the property holds of all natural numbers (Aczel 1977:742).

Mathematical Induction:

Piano's axioms

In our previous chapters, we were very careful when proving our various propositions and theorems to only use results we knew to be true. However, many of the statements that we take to be true had to be proven at some point. Those proofs, of course, relied on other true statements. If we continue to “trace back” our mathematics proofs, we begin to notice that mathematics must have some initial set of true statements that cannot be proven. These statements, known as axioms, are the starting point for any mathematical theory.

In this chapter, we will axiomatically define the natural numbers \mathbf{N} . As we move through the various axioms, we will see that each one is crucial in defining \mathbf{N} in such a way that they are equal to $\{0, 1, 2, 3, \dots\}$, which are the natural numbers that we know and love. After establishing this, we will establish the basic arithmetic operations on \mathbf{N} by defining addition and multiplication.

Axiomatizing the Natural Numbers

In this section, we will develop the Peano Axioms and use them to provide a completely formal definition of the natural numbers \mathbf{N} . In what follows, it is best to train yourself to assume nothing and use only statements that are known to be true via axioms or statements that follow from these axioms. We will formalize the notion of equality and then present the Peano axioms.

The Notion of Equality

When approaching mathematics axiomatically, it is important to not assume anything at all, including something as rudimentary as how equality behaves. After all, “=” is merely a symbol until we declare it to have some important properties. Below, we will define the

natural numbers \mathbb{N} axiomatically. Before we get deep into this, let's establish the properties that "=" should have. First, every natural number should be equal to itself; this is known as the reflexivity axiom.

Axiom 1. For every $x \in \mathbb{N}$, $x = x$

Next, if one natural number equals a second one, then that second one should equal the first one. This is called the symmetry axiom.

Axiom 2. For every $x, y \in \mathbb{N}$, if $x = y$, then $y = x$.

The next property allows us to say that if one natural number is equal to a second, and that second natural number is equal to a third, then the first and third are equal to each other. This is known as the transitivity axiom.

Axiom 3. For every $x, y, z \in \mathbb{N}$, if $x = y$ and $y = z$, then $x = z$

The above three properties of reflexivity, symmetry, and transitivity come up numerous times in essentially every mathematical field. Equality is an example of what is called a relation, and relations that enjoy the above three properties are known as equivalence relations.

There remains one last axiom related to equality. In each of the above axioms, whenever we used a symbol (like x , y , or z), we always had an assumption that these elements were in the natural numbers. If we don't make this assumption, then we may run into problems. To help get around this, the fourth axiom, called the closure of equality axiom, says that if you have a natural number that is equal to something, then that "something" also has to be a natural number.

Axiom 4. For all x and y , if $x \in \mathbb{N}$ and $x = y$, then $y \in \mathbb{N}$.

In other words, the only way for something to be equal to a natural number is for it to be a natural number itself.

In different versions of the Peano axioms, the above four axioms are excluded, as they these properties of equality are frequently assumed to be true as part of that logic system. For completeness, though, we include them in these notes. Furthermore, their inclusion here highlights the importance of questioning even the most basic mathematical assumptions.

The Peano Axioms

Now, we are ready to present the main Peano axioms. It is important to keep in mind that when Peano and others constructed these axioms, their goal was to provide the fewest axioms that would generate the natural numbers that everyone was familiar with.

The insight is that this could be done by asserting the existence of at least one natural number, and then defining a function, called successor function, that can be used to construct the remaining natural numbers

An obvious element to axiomatically include in the natural numbers is zero.

Axiom 5. 0 is a natural number. That is, $0 \in \mathbb{N}$.

In alternate versions of the Peano axioms, Axiom 5 actually replaces 0 with 1. This creates an almost identical set of natural numbers, which correspond to “positive whole numbers” (as we know them now). Whether a mathematician includes 0 in the natural numbers or not depends on the context. We use the convention of including 0 as a natural number.

At this point, we are only guaranteed the existence of a single natural number, 0. The next axiom uses the successor function to generate other natural numbers. As its name implies, the successor function is a function S that has as its domain \mathbb{N} . The next axiom simply states that the co-domain of S is also \mathbb{N} .

Axiom 6. If $x \in \mathbb{N}$, then $S(x) \in \mathbb{N}$. That is, if x is a natural number, then so its successor.

As the above axiom implies, we will commonly refer to $S(x)$ as the successor of x . Intuitively, we should think of $S(x)$ as $x+1$. Of course, we cannot formally define it this way yet since we do not know what $+$ means! At this point, we are still quite far away from having the natural numbers as we know them. For example, if we have $\mathbb{N} = \{0\}$ and define $S(0) = 0$, then all of the above axioms are satisfied. We, of course, want to avoid this. One way to ensure this is to insist that 0 is not the successor of any natural number (including itself, of course).

Axiom 7. For every natural number $x \in \mathbb{N}$, $S(x) = 0$ is false.

Rephrasing this using our knowledge of functions, we can say that the preimage of 0 under S in the natural numbers is the empty set. At this point, we are slightly closer to having \mathbb{N} look more like the natural numbers we know. In particular, we know that $S(0)$ is not equal to 0, and thus it must equal some other natural number. We can denote this natural number by 1. Thus, we can define 1 by $S(0) = 1$.

At this point, we know that \mathbb{N} contains at least two natural numbers, 0 and 1. If we were to stop here, though, we could not be guaranteed that all the rest of the natural numbers (as we know them) exist. For example, we could define $\mathbb{N} = \{0, 1\}$ where $S(0) = 1$ and $S(1) = 1$. Again, using our knowledge of functions, we recognize that the fact that $S(0) = 1$ and $S(1) = 1$ means that the S is not an injective function. We would like this successor function to be injective, so we have the following axiom.

Axiom 8. For all $x, y \in \mathbb{N}$, if $S(x) = S(y)$, then $x = y$.

The above axiom has some very important ramifications. First, it excludes the possibility of defining \mathbb{N} to be just $\{0, 1\}$. To see why, notice that we already have that $S(0) = 1$ and, by injectivity, we cannot have that $S(1) = 1$. Axiom 6 excludes the possibility that $S(1) = 0$. Thus, $S(1)$ must be some other natural number, which we denote as 2. Thus, we can define $2 = S(1)$. A similar argument gives that $S(2)$ cannot be 0, 1, or 2. Thus, it must be some other natural number, which we call 3. Continuing in this pattern, we see that \mathbb{N} must contain all the natural numbers that we know!

At this point, we have established that \mathbb{N} must include 0, its successor $1 = S(0)$, its successor's successor $2 = S(1)$, and so on. Thus, we formally have that \mathbb{N} must include $0, S(0), S(S(0)), S(S(S(0))), \dots$. Of course, to avoid so many nested applications of S , we use the numerals 1, 2, 3 to denote $S(0), S(S(0)),$ and $S(S(S(0)))$, respectively.

We are, however, not done. These first eight axioms have pushed our formal definition of \mathbb{N} to include all of the "usual" natural numbers that we know and love. That is, we know now that

$$\{0, 1, 2, \dots\} \subset \mathbb{N}.$$

However, what disallows our axiomatic \mathbb{N} from containing more? So far, nothing does. To see this, let's consider this version of \mathbb{N} that satisfies all the above axioms, but is not the usual natural numbers we know:

$$\mathbb{N} = \{0, 1, 2, 3, \dots\} \cup \{a, b\}$$

That is, this version of \mathbb{N} contains all the natural numbers and also includes two other symbols, a and b . We also need to describe the successor function. On the portion $\{0, 1, 2, 3, \dots\}$, we define S in the way described above, where $S(0) = 1, S(1) = 2, S(2) = 3,$ and so on. On the portion $\{a, b\}$, we can define $S(a) = b$ and $S(b) = a$. This version of \mathbb{N} with this successor function satisfies all the axioms, but is "larger" than we want our natural numbers to be. The next (and final) axiom will exclude versions of \mathbb{N} that are "too large" from occurring. Before we begin, we need a definition that is inspired by our previous chapter on induction. A set V is called inductive if the following two conditions are satisfied:

- $0 \in V$
- If $x \in V$, then $S(x) \in V$.

Of course, the name comes from the fact that the first condition is similar to our "base case" from induction, and the second condition is analogous to the induction step. The

last axiom, many times called the Axiom of Induction says that if V is an inductive set, then V contains the set of natural numbers.

Axiom 9. If V is an inductive set, then $\mathbb{N} \subset V$.

As stated above, the first 8 axioms ensure that $\{0, 1, 2, 3, \dots\} \subset \mathbb{N}$. Furthermore, notice that the set $\{0, 1, 2, 3, \dots\}$ is an inductive set! Thus, by Axiom 9, it must be true that $\mathbb{N} \subset \{0, 1, 2, 3, \dots\}$. Thus, we finally have the set equality that we were after:

$$\mathbb{N} = \{0, 1, 2, 3, \dots\}.$$

Mathematical Induction

Mathematical Induction (MI) is an extremely important tool in Mathematics.

First of all you should never confuse MI with Inductive Attitude in Science. The latter is just a process of establishing general principles from particular cases.

MI is a way of proving math statements for all integers (perhaps excluding a finite number) [1] says:

Statements proved by math induction all depend on an integer, say, n . For example,

$$(1) 1 + 3 + 5 + \dots + (2n - 1) = n^2$$

$$(2) \text{ If } x_1, x_2, \dots, x_n > 0 \text{ then } (x_1 + x_2 + \dots + x_n)/n \geq (x_1 \cdot x_2 \cdot \dots \cdot x_n)^{1/n}$$

etc. n here is an "arbitrary" integer.

It's convenient to talk about a statement $P(n)$. For (1), $P(1)$ says that $1 = 1^2$ which is incidentally true. $P(2)$ says that $1 + 3 = 2^2$, $P(3)$ means that $1 + 3 + 5 = 3^2$. And so on. These particular cases are obtained by substituting specific values 1, 2, 3 for n into $P(n)$.

Assume you want to prove that for some statement P , $P(n)$ is true for all n starting with $n = 1$. The Principle (or Axiom) of Math Induction states that, to this end, one should accomplish just two steps:

1. Prove that $P(1)$ is true.
2. Assume that $P(k)$ is true for some k . Derive from here that $P(k+1)$ is also true.

The idea of MI is that a finite number of steps may be needed to prove an infinite number of statements $P(1), P(2), P(3), \dots$

Let's prove (1). We already saw that $P(1)$ is true. Assume that, for an arbitrary k , $P(k)$ is also true, i.e. $1 + 3 + \dots + (2k-1) = k^2$. Let's derive $P(k+1)$ from this assumption. We have

$$\begin{aligned}
1 + 3 + \dots + (2k-1) + (2k+1) &= [1 + 3 + \dots + (2k-1)] + (2k+1) \\
&= k^2 + (2k+1) \\
&= (k+1)^2
\end{aligned}$$

Which exactly means that $P(k+1)$ holds. (For $2k+1 = 2(k+1)-1$.) Therefore, $P(n)$ is true for all n starting with 1.

Intuitively, the inductive (second) step allows one to say, look $P(1)$ is true and implies $P(2)$. Therefore $P(2)$ is true. But $P(2)$ implies $P(3)$. Therefore $P(3)$ is true which implies $P(4)$ and so on. Math induction is just a shortcut that collapses an infinite number of such steps into the two above.

In Science, inductive attitude would be to check a few first statements, say, $P(1)$, $P(2)$, $P(3)$, $P(4)$, and then assert that $P(n)$ holds for all n . The inductive step " $P(k)$ implies $P(k+1)$ " is missing. Needless to say nothing can be proved this way.

Remark

1. Often it is impractical to start with $n = 1$. MI applies with any starting integer n_0 . The result is then proved for all n from n_0 on.
2. Sometimes, instead of 2., one assumes 2':

Assume that $P(m)$ is true for all $m < (k + 1)$.

Derive from here that $P(k+1)$ is also true. The two approaches are equivalent, because one may consider statement Q : $Q(n) = P(1)$ and $P(2)$ and ... and $P(n)$, so that $Q(n)$ is true iff $P(1)$, $P(2)$, ..., $P(n)$ are all true.

This variant goes by the name of Complete Induction or Strong Induction.

In problem solving, mathematical induction is not only a means of proving an existing formula, but also a powerful methodology for finding such formulas in the first place. When used in this manner MI shows to be an outgrowth of (scientific) inductive reasoning - making conjectures on the basis of a finite set of observations.

Discrete Numeric Functions and Generating functions

Generating function is a method to solve the recurrence relations.

Let us consider, the sequence $a_0, a_1, a_2, \dots, a_r$ of real numbers. For some interval of real numbers containing zero values at t is given, the function $G(t)$ is defined by the series

$$G(t) = a_0 + a_1 t + a_2 t^2 + \dots + a_r t^r + \dots \text{equation (i)}$$

This function $G(t)$ is called the generating function of the sequence a_r .

Now, for the constant sequence 1, 1, 1, 1.....the generating function is

It can be expressed as

$$G(t) = (1-t)^{-1} = 1+t+t^2+t^3+t^4+\dots \text{ [By binomial expansion]}$$

Comparing, this with equation (i), we get

$$a_0=1, a_1=1, a_2=1 \text{ and so on.}$$

For, the constant sequence 1,2,3,4,5,..the generating function is

$$G(t) = \quad \text{because it can be expressed as}$$
$$G(t) = (1-t)^{-2} = 1+2t+3t^2+4t^3+\dots+(r+1)t^r$$

Comparing, this with equation (i), we get

$$a_0=1, a_1=2, a_2=3, a_3=4 \text{ and so on.}$$

The generating function of Z^r , ($Z \neq 0$ and Z is a constant) is given by

$$G(t) = 1+Zt+Z^2t^2+Z^3t^3+\dots+Z^r t^r$$

$$G(t) = \quad \text{[Assume } |Zt| < 1]$$

So, $G(t) = \quad$ generates $Z^r, Z \neq 0$

Also, if $a_r^{(1)}$ has the generating function $G_1(t)$ and $a_r^{(2)}$ has the generating function $G_2(t)$, then $\lambda_1 a_r^{(1)} + \lambda_2 a_r^{(2)}$ has the generating function $\lambda_1 G_1(t) + \lambda_2 G_2(t)$. Here λ_1 and λ_2 are constants.

Application Areas:

Generating functions can be used for the following purposes -

- For solving recurrence relations
- For proving some of the combinatorial identities
- For finding asymptotic formulae for terms of sequences

Example: Solve the recurrence relation $a_{r+2} - 3a_{r+1} + 2a_r = 0$

By the method of generating functions with the initial conditions $a_0=2$ and $a_1=3$.

Solution: Let us assume that

Multiply equation (i) by t^r and summing from $r = 0$ to ∞ , we have

$$(a_2 + a_3 t + a_4 t^2 + \dots) - 3(a_1 + a_2 t + a_3 t^2 + \dots) + 2(a_0 + a_1 t + a_2 t^2 + \dots) = 0$$

[$\therefore G(t) = a_0 + a_1 t + a_2 t^2 + \dots$]

$$+2G(t) = 0 \dots \dots \dots \text{equation (ii)}$$

Now, put $a_0 = 2$ and $a_1 = 3$ in equation (ii) and solving, we get

Put $t = 1$ on both sides of equation (iii) to find A. Hence

$$-1 = -A \quad \therefore A = 1$$

Put $t =$ on both sides of equation (iii) to find B. Hence

$$= B \quad \therefore B = 1$$

Thus $G(t) =$. Hence, $a_r = 1 + 2^r$.

Simple Recurrence relation with constant coefficients

A Recurrence Relations is called linear if its degree is one.

The general form of linear recurrence relation with constant coefficient is

$$C_0 y_{n+r} + C_1 y_{n+r-1} + C_2 y_{n+r-2} + \dots + C_r y_n = R(n)$$

Where $C_0, C_1, C_2, \dots, C_n$ are constant and $R(n)$ is same function of independent variable n .

A solution of a recurrence relation in any function which satisfies the given equation.

Linear Homogeneous Recurrence Relations with Constant Coefficients:

The equation is said to be linear homogeneous difference equation if and only if $R(n) = 0$ and it will be of order n .

The equation is said to be linear non-homogeneous difference equation if $R(n) \neq 0$.

Example1:

The equation $a_{r+3} + 6a_{r+2} + 12a_{r+1} + 8a_r = 0$ is a linear non-homogeneous equation of order 3.

Example2:

The equation $a_{r+2} - 4a_{r+1} + 4a_r = 3r + 2^r$ is a linear non-homogeneous equation of order 2.

A linear homogeneous difference equation with constant coefficients is given by

$$C_0 y_n + C_1 y_{n-1} + C_2 y_{n-2} + \dots + C_r y_{n-r} = 0 \dots \dots \text{equation (i)}$$

Where $C_0, C_1, C_2, \dots, C_n$ are constants.

The solution of the equation (i) is of the form $A \alpha_1^k$, where α_1 is the characteristics root and A is constant.

Substitute the values of $A \alpha^k$ for y_n in equation (1), we have

$$C_0 A \alpha^k + C_1 A \alpha^{k-1} + C_2 A \alpha^{k-2} + \dots + C_r A \alpha^{k-r} = 0 \dots \dots \text{equation (ii)}$$

After simplifying equation (ii), we have

$$C_0 \alpha^r + C_1 \alpha^{r-1} + C_2 \alpha^{r-2} + \dots + C_r = 0 \dots \dots \text{equation (iii)}$$

The equation (iii) is called the characteristics equation of the difference equation.

If α_1 is one of the roots of the characteristics equation, then $A \alpha_1^k$ is a homogeneous solution to the difference equation.

To find the solution of the linear homogeneous difference equations, we have the four cases that are discussed as follows:

Case1:

If the characteristic equation has n distinct real roots $\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n$.

Thus, $e^{\alpha_k k}$ are all solutions of equation (i).

Also, we have $e^{(\alpha_1 + \alpha_2)k}$ are all solutions of equation (i). The sums of solutions are also solutions.

Hence, the homogeneous solutions of the difference equation are

Case2:

If the characteristic equation has repeated real roots.

If $\alpha_1 = \alpha_2$, then $(A_1 + A_2 k) e^{\alpha_1 k}$ is also a solution.

If $\alpha_1 = \alpha_2 = \alpha_3$ then $(A_1 + A_2 k + A_3 k^2) e^{\alpha_1 k}$ is also a solution.

Similarly, if root α_1 is repeated n times, then.

$$(A_1 + A_2 k + A_3 k^2 + \dots + A_n k^{n-1}) e^{\alpha_1 k}$$

The solution to the homogeneous equation.

Case3:

If the characteristic equation has one imaginary root.

If $\alpha + i\beta$ is the root of the characteristic equation, then $\alpha - i\beta$ is also the root, where α and β are real.

Thus, $(\alpha + i\beta)^k$ and $(\alpha - i\beta)^k$ are solutions of the equations. This implies

$$(\alpha + i\beta)^k A_1 + (\alpha - i\beta)^k A_2$$

is also a solution to the characteristic equation, where A_1 and A_2 are constants which are to be determined.

Case4:

If the characteristics equation has repeated imaginary roots.

When the characteristics equation has repeated imaginary roots,

$$(C_1+C_2 k) (\alpha+i\beta)^k +(C_3+C_4 K)(\alpha-i\beta)^k$$

Is the solution to the homogeneous equation.

Example1:

Solve the difference equation $a_r-3a_{r-1}+2a_{r-2}=0$.

Solution:

The characteristics equation is given by

$$s^2-3s+2=0 \text{ or } (s-1)(s-2)=0$$

$$\Rightarrow s = 1, 2$$

Therefore, the homogeneous solution of the equation is given by

$$a_r=C^1_r+C^2_r.2^r.$$

Example2:

Solve the difference equation $9y_{k+2}-6y_{k+1}+y_k=0$.

Solution:

The characteristics equation is

$$9s^2-6s+1=0 \text{ or } (3s-1)^2=0$$

$$\Rightarrow s = \quad \text{and}$$

Therefore, the homogeneous solution of the equation is given by

$$y_k=(C_1+C_2 k).$$

Example3:

Solve the difference equation $y_k-y_{k-1}-y_{k-2}=0$.

Solution:

The characteristics equation is $s^2-s-1=0$

$s=$

Therefore, the homogeneous solution of the equation is

Example4:

Solve the difference equation $y_{k+4}+4y_{k+3}+8y_{k+2}+8y_{k+1}+4y_k=0$.

Solution:

The characteristics equation is $s^4+4s^3+8s^2+8s+4=0$

$$(s^2+2s+2)(s^2+2s+2)=0$$

$$s = -1 \pm i, -1 \pm i$$

Therefore, the homogeneous solution of the equation is given by

$$y_k=(C_1+C_2 K)(-1+i)^k+(C_3 +C_4 K)(-1-i)^k$$

Asymptotic Behavior of functions

Consider two types of extreme (or asymptotic) registrations:

- large translations -- such that the volume overlap is minimal.
- large scale disparity -- such that a small portion of the floating volume is stretched to cover the entire reference volume.

Both of these cases represent poor registrations and it is therefore important that the cost values associated with them are high. If this is not the case then the global minimum may be associated with these transformations rather than the desired one, violating the first assumption outlined in section 2.5. Furthermore, limiting the domain to exclude these transformations will only eliminate the problem if the cost at the domain boundary is guaranteed to be larger than the global minimum, which is generally difficult or impossible to do as the value at the global minimum is unknown.

In examining the asymptotic behaviour it is important to define what the boundary conditions are. That is, what is done for points in space which do not lie in one or other of the volume domains. One option is to (conceptually) pad the volumes with zeros to give them infinite domain. However, this creates artificial intensity boundaries when the FOV only includes part of the head (for example, when the top few mm of the

head/brain are not included the intensity suddenly drops from that of brain matter to zero) -- which is relatively common. These artificial boundaries would then bias the registration, which is undesirable. Therefore, a better alternative is to only calculate the cost function for the region where the volumes overlap. This usually requires some extra calculations in practice, as the normalisation now depends on the overlap volume, which depends on the transformation, but the resulting registration is, in theory, unbiased.

The cost functions that will be compared here are: the Woods function [Woods et al., 1993], Correlation Ratio [Roche et al., 1998], Joint Entropy [Studholme et al., 1995, Collignon et al., 1995], Mutual Information [Viola and Wells, 1997, Maes et al., 1997], and Normalised Mutual Information [Studholme et al., 1999, Maes, 1998].

Denoting the two images by X (typically I') and Y (typically I''), the respective cost functions¹ are defined as:

$$C^W = \frac{\sum_{i,j} (X_i - \mu_X)^2 (Y_j - \mu_Y)^2}{\sum_{i,j} (X_i - \mu_X)^2 \sum_{i,j} (Y_j - \mu_Y)^2} \quad (7)$$

$$C^{CR} = \frac{\sum_{i,j} (X_i - \mu_X)^2 (Y_j - \mu_Y)^2}{\sum_{i,j} (X_i - \mu_X)^2 \sum_{i,j} (Y_j - \mu_Y)^2} \quad (8)$$

$$C^{JE} = H(X, Y) \quad (9)$$

$$C^{MI} = H(X, Y) - H(X) - H(Y) \quad (10)$$

$$C^{NMI} = \frac{H(X, Y)}{H(X) + H(Y)} \quad (11)$$

Here the quantities X and Y represent the images as the set of intensities evaluated at the discrete, valid grid points. That is, where G represents the discrete grid and the continuous, valid domain (that is, the FOV). Also:

- μ_A is the mean of set A
- σ_A^2 is the variance of the set A
- Y_i is the ith iso-set defined by X as
 - $n_i = \text{Card} (Y_i)$ (number of elements in the set Y_i) such that
 - $p_{i,j}$ is the standard entropy definition where $p_{i,j}$ represents the probability of the (i,j) joint histogram bin, and similarly for the marginals, $H(X)$ and $H(Y)$.

These definition require the specification of a partition of the intensities:

. This partition is used to define the various histograms and iso-sets required for the calculation of the cost functions. In particular, a discrete bin (or iso-set) number is calculated for each voxel using the intensity at that voxel. For example, at location q in

image X the intensity is $X(q)$ and the bin number is k if $I_k < X(q) < I_{k+1}$. Then, given this bin number, iso-sets or the joint histogram are easily determined.

That is, the k th iso-set, Y_k , is the set of Y intensities where the corresponding voxel in X has a bin number k -- the intensity $X(q)$ is between I_k and I_{k+1} . The joint histogram is composed of a number of elements, where the element $b(i,j)$ represents the number of voxels where the intensity of X is in the i th bin, (I_i, I_{i+1}) , and the intensity of Y for the same (corresponding) voxel is in the j th bin, (I_j, I_{j+1}) . The probability associated with this element p_{ij} is then simply the value of the element divided by the sum of all the elements.

Algebraic Structures:

Properties

A non-empty set G equipped with one or more binary operations is said to be an algebraic structure. Suppose $*$ is a binary operation on G . Then $(G, *)$ is an algebraic structure. $(\mathbb{N}, *)$, $(\mathbb{1}, +)$, $(\mathbb{1}, -)$ are all the algebraic structure. Here, $(\mathbb{R}, +, \cdot)$ is an algebraic structure equipped with two operations.

Binary operation on a set

Suppose G is a non-empty set. The $G \times G = \{(a,b) : a \in G, b \in G\}$. If $f : G \times G \rightarrow G$ then f is called a binary operation on a set G . The image of the ordered pair (a,b) under the function f is denoted by afb .

A binary operation on a set G is sometimes also said to be the binary composition in the set G . If $*$ is a binary composition in G then, $a * b \in G$, $a, b \in G$. Therefore G is closed with respect to the composition denoted by $*$.

Example:

Addition is a binary operation on the set \mathbb{N} of natural number. The sum of two natural number is also a natural number. Therefore, \mathbb{N} is a natural number with respect to addition i.e. $a+b$.

Subtraction is not a binary operation on \mathbb{N} . We have $4 - 7 = 3$ not belong to \mathbb{N} whereas $4 \in \mathbb{N}$. thus, \mathbb{N} is not closed with respect to subtraction, but subtraction is a binary operation on the set of an integer.

Properties of an algebraic structure

By a property of an algebraic structure, we mean a property possessed by any of its operations. Important properties of an algebraic system are:

1. Associative and commutative laws

An operation $*$ on a set is said to be associative or to satisfy the associative law if, for any elements a, b, c in S we have $(a * b) * c = a * (b * c)$

An operation $*$ on a set S is said to be commutative or satisfy the commutative law if, $a * b = b * a$ for any element a, b in S .

2. Identity element and inverse

Consider an operation $*$ on a set S . An element e in S is called an identity element for $*$ if for any elements a in S - $a * e = e * a = a$

Generally, an element e is called a left identity or a right identity according to as $e * a$ or $a * e = a$ where a is any elements in S .

Suppose an operation $*$ on a set S does have an identity element e . The inverse of an element in S is an element b such that: $a * b = b * a = e$

3. Cancellation laws

An operation $*$ on a set S is said to satisfy the left cancellation law if, $a * b = a * c$ implies $b = c$ and is said to satisfy the right cancellation law if, $b * a = c * a$ implies $b = c$

Semi group

Let us consider, an algebraic system $(A, *)$, where $*$ is a binary operation on A . Then, the system $(A, *)$ is said to be semi-group if it satisfies the following properties:

1. The operation $*$ is a closed operation on set A .
2. The operation $*$ is an associative operation.

Example: Consider an algebraic system $(A, *)$, where $A = \{1, 3, 5, 7, 9, \dots\}$, the set of positive odd integers and $*$ is a binary operation means multiplication. Determine whether $(A, *)$ is a semi-group.

Solution: Closure Property: The operation $*$ is a closed operation because multiplication of two +ve odd integers is a +ve odd number.

Associative Property: The operation $*$ is an associative operation on set A . Since every $a, b, c \in A$, we have

$$(a * b) * c = a * (b * c)$$

Hence, the algebraic system $(A, *)$, is a semigroup.

Subsemigroup:

Consider a semigroup $(A, *)$ and let $B \subseteq A$. Then the system $(B, *)$ is called a subsemigroup if the set B is closed under the operation $*$.

Example: Consider a semigroup $(\mathbb{N}, +)$, where \mathbb{N} is the set of all natural numbers and $+$ is an addition operation. The algebraic system $(E, +)$ is a subsemigroup of $(\mathbb{N}, +)$, where E is a set of +ve even integers.

Free Semigroup:

Consider a non empty set $A = \{a_1, a_2, \dots, a_n\}$.

Now, A^* is the set of all finite sequences of elements of A , i.e., A^* consist of all words that can be formed from the alphabet of A .

If α, β , and γ are any elements of A^* , then $\alpha(\beta \cdot \gamma) = (\alpha \cdot \beta) \cdot \gamma$.

Here \circ is a concatenation operation, which is an associative operation as shown above.

Thus (A^*, \circ) is a semigroup. This semigroup (A^*, \circ) is called the free semigroup generated by set A .

Product of Semigroup:

Theorem: If $(S_1, *)$ and $(S_2, *)$ are semigroups, then $(S_1 \times S_2, *)$ is a semigroup, where $*$ defined by $(s_1', s_2') * (s_1'', s_2'') = (s_1' * s_1'', s_2' * s_2'')$.

Proof: The semigroup $S_1 \times S_2$ is closed under the operation $*$.

Associativity of $*$. Let $a, b, c \in S_1 \times S_2$

$$\begin{aligned} \text{So, } a * (b * c) &= (a_1, a_2) * ((b_1, b_2) * (c_1, c_2)) \\ &= (a_1, a_2) * (b_1 * c_1, b_2 * c_2) \\ &= (a_1 * (b_1 * c_1), a_2 * (b_2 * c_2)) \\ &= ((a_1 * b_1) * c_1, (a_2 * b_2) * c_2) \\ &= (a_1 * b_1, a_2 * b_2) * (c_1, c_2) \\ &= ((a_1, a_2) * (b_1, b_2)) * (c_1, c_2) \\ &= (a * b) * c. \end{aligned}$$

Since $*$ is closed and associative. Hence, $S_1 \times S_2$ is a semigroup.

Monoid:

Let us consider an algebraic system (A, o) , where o is a binary operation on A . Then the system (A, o) is said to be a monoid if it satisfies the following properties:

1. The operation o is a closed operation on set A .
2. The operation o is an associative operation.
3. There exists an identity element, i.e., the operation o .

Example: Consider an algebraic system $(N, +)$, where the set $N = \{0, 1, 2, 3, 4, \dots\}$. The set of natural numbers and $+$ is an addition operation. Determine whether $(N, +)$ is a monoid.

Solution: (a) Closure Property: The operation $+$ is closed since the sum of two natural numbers.

(b) Associative Property: The operation $+$ is an associative property since we have $(a+b)+c=a+(b+c) \forall a, b, c \in N$.

(c) Identity: There exists an identity element in set N the operation $+$. The element 0 is an identity element, i.e., the operation $+$. Since the operation $+$ is a closed, associative and there exists an identity. Hence, the algebraic system $(N, +)$ is a monoid.

SubMonoid:

Let us consider a monoid (M, o) , also let $S \subseteq M$. Then (S, o) is called a submonoid of (M, o) , if and only if it satisfies the following properties:

1. S is closed under the operation o .
2. There exists an identity element $e \in T$.

Example: Let us consider, a monoid $(M, *)$, where $*$ is a binary operation and M is a set of all integers. Then $(M_1, *)$ is a submonoid of $(M, *)$ where M_1 is defined as $M_1 = \{a^i \mid i \text{ is from } 0 \text{ to } n, a \text{ positive integer, and } a \in M\}$.

Monoid

- A group is a monoid with an inverse element. The inverse element (denoted by I) of a set S is an element such that $(a \circ I) = (I \circ a) = a$, for each element $a \in S$.
- So, a group holds four properties simultaneously –
 1. Closure,
 2. Associative,
 3. Identity element,
 4. inverse element.

- The order of a group G is the number of elements in G and the order of an element in a group is the least positive integer n such that a^n is the identity element of that group G .

Examples

- The set of $N \times N$ non-singular matrices form a group under matrix multiplication operation.
- The product of two $N \times N$ non-singular matrices is also an $N \times N$ non-singular matrix which holds closure property.
- Matrix multiplication itself is associative. Hence, associative property holds.
- The set of $N \times N$ non-singular matrices contains the identity matrix holding the identity element property.
- As all the matrices are non-singular they all have inverse elements which are also non-singular matrices. Hence, inverse property also holds.

Monoid:

If a semigroup $\{M, *\}$ has an identity element with respect to the operation $*$, then $\{M, *\}$ is called a *monoid*.

viz., if for any $a, b, c \in M$

$$(a*b)*c = a*(b*c)$$

and if there exists an element $e \in M$ such that for

any $a \in M, e*a = a*e = a$, then the algebraic system $\{M, *\}$ is called a monoid.

For example, if N is the set of natural numbers, then $\{N, +\}$ and $\{N, X\}$ are monoids with the identity elements 0 and 1 respectively.

The semigroups $\{E, +\}$ and $\{E, X\}$ are not monoids.

Semigroup:

If S is a nonempty set and $*$ be a binary operation on S , then the algebraic system $\{S, *\}$ is called a *semigroup*, if the operation $*$ is associative.

viz., if for any $a, b, c \in S$,

$$(a*b)*c = a*(b*c)$$

Since the characteristic property of a binary operation on S is the closure property, it is not necessary to mention it explicitly when algebraic systems are defined.

For example, if E is the set of positive even numbers, then $\{E, +\}$ and $\{E, X\}$ are semigroups.

Abelian Group

We no longer assume that the groups we study are finite.

With abelian groups, additive notation is often used instead of multiplicative notation. In other words the identity is represented by 0 , and $a+b$ represents the element obtained from applying the group operation to a and b .

A group G is the direct sum of two subgroups U, V if every element $x \in G$ can be written in the form $x = u + v$ where $u \in U, v \in V$, and $u + v = 0$ implies $u = v = 0$. We write $G = U \oplus V$.

Note that U, V cannot have a nonzero element w in common, otherwise $w + (-w) = 0$ is a nontrivial decomposition of zero. Also u, v are uniquely determined by x if $u_1 + v_1 = u_2 + v_2$ implies $u_1 - u_2 = v_2 - v_1 \in U \cap V$.

More generally we have $G = U_1 \oplus \dots \oplus U_r$, if every $x \in G$ can be written in the form $x = u_1 + \dots + u_r$ and also $0 = u_1 + \dots + u_r$ implies $0 = u_1 = \dots = u_r$. Clearly if G is finite we have $|G| = |U_1| \dots |U_r|$.

An abelian group A is a free abelian group of rank r if there exist $u_1, \dots, u_r \in A$ such that $A = \langle u_1, \dots, u_r \rangle$ and $a_1 u_1 + \dots + a_r u_r = 0$ implies $a_1 = \dots = a_r = 0$. Alternatively we may require every $x \in A$ can be uniquely written in the form $x = a_1 u_1 + \dots + a_r u_r$. The set $\{u_1, \dots, u_r\}$ is a set of free generators of A . The trivial group is viewed as a free abelian group of rank zero, and viewed as been generated by the empty set.

Generators need not be unique. However it is easy to see that two sets of free generators are related by a unimodular (determinant of absolute value one) matrix transformation.

Theorem: [Dedekind] Let F be a free abelian group of rank r and let G be a nonzero subgroup of F . Then G is a free abelian group of rank s with $s \leq r$. Furthermore, F has a set of free generators $\{u_1, \dots, u_r\}$ such that G is generated by

$v_1 = a_{11}u_1 + a_{12}u_2 + \dots + a_{1r}u_r$
 $v_2 = a_{21}u_1 + a_{22}u_2 + \dots + a_{2r}u_r$
 \vdots
 $v_s = a_{s1}u_1 + a_{s2}u_2 + \dots + a_{sr}u_r$
 \vdots
 $v_r = a_{r1}u_1 + a_{r2}u_2 + \dots + a_{rr}u_r$
 for some a_{ij} with $a_{11}, a_{22}, \dots, a_{ss}$ positive.

Proof: Let $\{u_1, \dots, u_r\}$ be free generators for F . Then take any nonzero element $b = b_1 u_1 + \dots + b_r u_r$ of G . After permuting the u_i 's if necessary, assume $b_1 \neq 0$. Then since G is closed under inverses, we may take $b_1 > 0$.

Enumerate all elements $x_1u_1+\dots+x_rur$ of G and consider the set of possible positive integer values for x_1 . We know this set is nonempty since b_1 is a possible value. Then call the smallest integer in this set a_1 and take any element $v_1=a_1u_1+\dots+a_1rur \in G$ for which this minimum is attained.

Then every element $x_1u_1+\dots+x_rur \in G$ must satisfy $a_1|x_1$, since we have $x_1=a_1q+bx_1=a_1q+b$ for integers q, b with $0 \leq b < a_1$ (which implies $x_1=b$ for some element of G), and we have chosen a_1 to be minimal.

Thus for all $x \in G$, for some integer q we have $x - qv_1 = b_2u_2 + \dots + b_rur$ for some b_2, \dots, b_r . If $r=1$ then we are done since we have $F = \langle u_1 \rangle, G = \langle a_1u_1 \rangle$.

We use induction. Suppose $r > 1$.

Let $F_1 = \langle u_2, \dots, u_r \rangle, G_1 = G \cap F_1$. Then G_1 is a subgroup of F_1 and by inductive hypothesis $G_1 = \langle v_2, \dots, v_s \rangle$ where $s \leq r$ and

$v_2 = a_{22}u_2 + a_{23}u_3 + \dots + a_{2r}ur, v_3 = a_{33}u_3 + \dots + a_{3r}ur; \dots, v_s = a_{ss}u_s + \dots + a_{sr}ur$
with a_{22}, \dots, a_{ss} positive. We claim v_1, \dots, v_s generate G . We have already seen that for any $x \in G$, there exists some integer q such that $x - qv_1 \in F_1$. Then $x - qv_1 \in G_1$, hence $G = \langle v_1, \dots, v_s \rangle$.

It remains to show that v_1, \dots, v_s are independent. Suppose not, that is, there exists a nontrivial relation $c_1v_1 + \dots + c_s v_s = 0$. We must have $c_1 \neq 0$ because by induction we cannot have a nontrivial relation between v_2, \dots, v_s . Expressing the v_i 's in terms of the u_i 's, we arrive at a nontrivial relation between the u_i 's since the coefficient of u_1 is $c_1 a_{11} \neq 0$, a contradiction since the u_i 's are independent. ■

Now let $F = \langle u_1, \dots, u_r \rangle$ be an abelian free group of rank r . Recall any set of generators of F is related to the u_i 's via a unimodular matrix transformation, hence such a generator $b_1u_1 + \dots + b_rur$ must have $\gcd(b_1, \dots, b_r) = 1$. The converse is also true:

Lemma: Let $F = \langle u_1, \dots, u_r \rangle$. Let $v = b_1u_1 + \dots + b_rur$ with $\gcd(b_1, \dots, b_r) = 1$. Then there exist $v_2, \dots, v_r \in F$ with $F = \langle v, v_2, \dots, v_r \rangle$.

Proof: Set $s = |b_1| + \dots + |b_r|$. If $s=1$ then the result is trivial, since we have $v = \pm u_i$ for some i . We shall induct on s .

If $s > 1$ then at least two of the b_i 's are nonzero, and without loss of generality assume $b_1 \geq b_2 > 0$. Then

set $u'_1 = u_1, u'_2 = u_1 + u_2, u'_j = u_j$ for $j \geq 3$.
 Clearly $F = \langle u'_1, \dots, u'_r \rangle$, and we have

$v = (b_1 - b_2)u'_1 + b_2u'_2 + \dots + br'u'_r = (b_1 - b_2)u_1 + b_2u_2 + \dots + br'u_r$
 Furthermore $\gcd(b_1 - b_2, b_2, \dots, br) = 1$ and

$|b_1 - b_2| + |b_2| + \dots + |br| < s$
 so by inductive hypothesis the result follows. ■

Theorem: Let F be a finitely generated free abelian group of rank r and let G be a subgroup of F of rank s with $0 < s \leq r$. Then there exist generators for F v_1, \dots, v_r such that

$G = \langle h_1v_1, \dots, h_s v_s \rangle$
 where h_1, \dots, h_s are positive integers satisfying $h_i | h_{i+1}$ for $i = 1, \dots, s-1$.

Proof: Let u_1, \dots, u_r be a set of generators for F . Take any $x \in G$. Write $x = x_1u_1 + \dots + x_ru_r$. Define $\delta(x) = \gcd(x_1, \dots, x_r)$. We claim that $\delta(x)$ is independent of the choice of generators of F .

This is easily seen because if u'_1, \dots, u'_r are another set of generators, we can write the u_i 's in terms of the u'_i 's showing that $\gcd(x_1, \dots, x_r) | \gcd(x'_1, \dots, x'_r)$ where $x = x'_1u'_1 + \dots + x'_ru'_r$. By symmetry we must have equality.

Now take any nonzero $y_1 \in G$ such that $\delta(y_1)$ is minimal. Set $h_1 = \delta(y_1)$. Then y_1 can be written $y_1 = h_1(z_1u_1 + \dots + z_ru_r)$ for some integers z_i satisfying $\gcd(z_1, \dots, z_r) = 1$. By the lemma, there exist elements v_2, \dots, v_r which together with v_1 generate F .

Hence an element $y \in G$ can be written

$y = w_1v_1 + w_2v_2 + \dots + w_rv_r$
 Now h_1 must divide w_1 , since we have $w_1 = qh_1 + mw_1 = qh_1 + m$ for some $0 \leq q < h_1$ and h_1 is minimal. (Consider $\delta(y - qy_1)$.) Thus

$y - qy_1 = t_2v_2 + \dots + t_rv_r$
 If $r=1$ we are done, for we have $s=1, F = \langle v_1 \rangle, G = \langle h_1v_1 \rangle$. We induct on r , so suppose $r > 1$.

Let $F_1 = \langle v_1, v_2, \dots, v_r \rangle$ and $G_1 = F_1 \cap G$. Then G_1 is a subgroup of F_1 whose rank we shall denote by $t-1$ where $0 < t \leq r$. If $t=1$ then $G_1 = 0$ and since $G = \langle h_1v_1 \rangle$ we are done.

Otherwise $t < 1$, and by inductive hypothesis there exist free generators v_2, \dots, v_r of F_1 such that

$$G_1 = \langle h_2 v_2, \dots, h_t v_t \rangle$$

where $h_i | h_{i+1}$ for $i = 2, \dots, t-1$. Now $F = \langle v_1, \dots, v_r \rangle$ and any $y \in G$ can be written $y = q_1 h_1 v_1 + g_1$ for some $g_1 \in G_1$. Thus $h_1 v_1, \dots, h_t v_t$ generate G . They must also be independent, because a nontrivial relation between them imply a nontrivial relation between the generators v_1, \dots, v_r of F .

Thus $G = \langle h_1 v_1, \dots, h_t v_t \rangle$ and $t = s$. It remains to show $h_1 | h_2$. Write $h_2 = a h_1 + b$ where $0 \leq b < h_1$. Then consider $y_0 = h_1 v_1 + h_2 v_2 \in G$. We have $\delta(y_0) = \gcd(h_1, h_2) = \gcd(h_1, b)$. By minimality of h_1 we must have $b = 0$.

properties of group

- The identity element of a group is unique.
- The inverse of each element of a group is unique, i.e. in a group G with operation $*$ for every $a \in G$, there is only element a^{-1} such that $a^{-1} * a = a * a^{-1} = e$, e being the identity.
- The inverse of a^{-1} is a , i.e. $(a^{-1})^{-1} = a$.
- The inverse of the product of two elements of a group G is the product of the inverse taken in the inverse order, i.e. $(a * b)^{-1} = b^{-1} * a^{-1} \forall a, b \in G$.
- Cancellation laws holds in a group, i.e. if a, b, c are any elements of a group G , then $a * b = a * c \Rightarrow b = c$ (left cancellation law), $b * a = c * a \Rightarrow b = c$ (right cancellation law).
- If G is a group with binary operation $*$ and if a and b are any elements of G , then the linear equations $a * x = b$ and $y * a = b$ have unique solutions in G .
- The left inverse of an element is also its right inverse, i.e. $a^{-1} * a = e = a * a^{-1}$.

Subgroup

If a non-void subset H of a group G is itself a group under the operation of G , we say H is a subgroup of G .

Theorem: - A subset H of a group G is a subgroup of G if:

- the identity element $a \in H$.
- H is closed under the operation of G i.e. if $a, b \in H$, then $a, b \in H$ and
- H is closed under inverses, that is if $a \in H$ then $a^{-1} \in H$.

Cyclic Subgroup:-

A Subgroup K of a group G is said to be cyclic subgroup if there exists an element $x \in G$ such that every element of K can be written in the form x^n for some $n \in \mathbb{Z}$.

The element x is called generator of K and we write $K = \langle x \rangle$

Cyclic Group:-

In the case when $G = K$, we say G is cyclic and x is a generator of G . That is, a group G is said to be cyclic if there is an element $x \in G$ such that every element of G can be written in the form x^n for the some $n \in \mathbb{Z}$.

Example: The group $G = \{1, -1, i, -i\}$ under usual multiplication is a finite cyclic group with i as generator, since $i^1 = i, i^2 = -1, i^3 = -i$ and $i^4 = 1$

Abelian Group:

Let us consider an algebraic system $(G, *)$, where $*$ is a binary operation on G . Then the system $(G, *)$ is said to be an abelian group if it satisfies all the properties of the group plus a additional following property:

(1) The operation $*$ is commutative i.e.,
 $a * b = b * a \forall a, b \in G$

Example: Consider an algebraic system $(G, *)$, where G is the set of all non-zero real numbers and $*$ is a binary operation defined by

Show that $(G, *)$ is an abelian group.

Solution:

Closure Property: The set G is closed under the operation $*$, since $a * b =$ is a real number. Hence, it belongs to G .

Associative Property: The operation $*$ is associative. Let $a, b, c \in G$, then we have

Identity: To find the identity element, let us assume that e is a +ve real number. Then $e * a = a$, where $a \in G$.

Thus, the identity element in G is 4.

Inverse: let us assume that $a \in G$. If $a^{-1} \in Q$, is an inverse of a , then $a * a^{-1} = 4$

Thus, the inverse of element a in G is

Commutative: The operation $*$ on G is commutative.

Thus, the algebraic system $(G, *)$ is closed, associative, identity element, inverse and commutative. Hence, the system $(G, *)$ is an abelian group.

Product of Groups:

Theorem: Prove that if $(G_1, *_1)$ and $(G_2, *_2)$ are groups, then $G = G_1 \times G_2$ i.e., $(G, *)$ is a group with operation defined by $(a_1, b_1) * (a_2, b_2) = (a_1 *_1 a_2, b_1 *_2 b_2)$.

Proof: To prove that $G_1 \times G_2$ is a group, we have to show that $G_1 \times G_2$ has the associativity operator, has an identity and also exists inverse of every element.

Associativity. Let $a, b, c \in G_1 \times G_2$, then

$$\begin{aligned} \text{So, } a * (b * c) &= (a_1, a_2) * ((b_1, b_2) * (c_1, c_2)) \\ &= (a_1, a_2) * (b_1 *_1 c_1, b_2 *_2 c_2) \\ &= (a_1 *_1 (b_1 *_1 c_1), a_2 *_2 (b_2 *_2 c_2)) \\ &= ((a_1 *_1 b_1) *_1 c_1, (a_2 *_2 b_2) *_2 c_2) \\ &= (a_1 *_1 b_1, a_2 *_2 b_2) * (c_1, c_2) \\ &= ((a_1, a_2) * (b_1, b_2)) * (c_1, c_2) \\ &= (a * b) * c. \end{aligned}$$

Identity: Let e_1 and e_2 are identities for G_1 and G_2 respectively. Then, the identity for $G_1 \times G_2$ is $e = (e_1, e_2)$. Assume same $a \in G_1 \times G_2$

$$\begin{aligned} \text{Then, } a * e &= (a_1, a_2) * (e_1, e_2) \\ &= (a_1 *_1 e_1, a_2 *_2 e_2) \\ &= (a_1, a_2) = a \end{aligned}$$

Similarly, we have $e * a = a$.

Inverse: To determine the inverse of an element in $G_1 \times G_2$, we will determine it component wise i.e.,

$$a^{-1} = (a_1, a_2)^{-1} = (a_1^{-1}, a_2^{-1})$$

Now to verify that this is the exact inverse, we will compute $a * a^{-1}$ and $a^{-1} * a$.

$$\begin{aligned} \text{Now, } a * a^{-1} &= (a_1, a_2) * (a_1^{-1}, a_2^{-1}) \\ &= (a_1 * a_1^{-1}, a_2 * a_2^{-1}) = (e_1, e_2) = e \end{aligned}$$

Similarly, we have $a^{-1} * a = e$.

Thus, $(G_1 \times G_2, *)$ is a group.

In general, if G_1, G_2, \dots, G_n are groups, then $G = G_1 \times G_2 \times \dots \times G_n$ is also a group.

Cosets:

Let H be a subgroup of a group G . A left coset of H in G is a subset of G whose elements may be expressed as $xH = \{xh \mid h \in H\}$ for any $x \in G$. The element x is called a representation of the coset. Similarly, a right coset of H in G is a subset that may be expressed as $Hx = \{hx \mid h \in H\}$, for any $x \in G$. Thus complexes xH and Hx are called respectively a left coset and a right coset.

If the group operation is additive (+) then a left coset is denoted as $x + H = \{x+h \mid h \in H\}$ and a right coset is denoted by $H + x = \{h+x \mid h \in H\}$

Cyclic group

In group theory, a branch of abstract algebra, a cyclic group or monogenous group is a group that is generated by a single element.^[1] That is, it is a set of invertible elements with a single associative binary operation, and it contains an element g such that every other element of the group may be obtained by repeatedly applying the group operation to g or its inverse. Each element can be written as a power of g in multiplicative notation, or as a multiple of g in additive notation. This element g is called a generator of the group.^[1]

Every infinite cyclic group is isomorphic to the additive group of \mathbb{Z} , the integers. Every finite cyclic group of order n is isomorphic to the additive group of $\mathbb{Z}/n\mathbb{Z}$, the integers modulo n . Every cyclic group is an abelian group (meaning that its group operation is commutative), and every finitely generated abelian group is a direct product of cyclic groups.

Every cyclic group of prime order is a simple group which cannot be broken down into smaller groups. In the classification of finite simple groups, one of the three infinite classes consists of the cyclic groups of prime order. The cyclic groups of prime order are thus among the building blocks from which all groups can be built.

Cosets

Coset is subset of mathematical group consisting of all the products obtained by multiplying fixed element of group by each of elements of given subgroup, either on right or on left. Cosets are basic tool in study of groups
Suppose if A is group, and B is subgroup of A , and a is an element of A , then

$aB = \{ab : b \text{ an element of } B\}$ is left coset of B in A ,

The left coset of B in A is subset of A of form aB for some a (element of A). In aB (left coset), a is representative of coset.
and

$Ba = \{ba : b \text{ an element of } B\}$ is right coset of B in A .

The right coset of B in A is subset of A of form Ba for some a (element of A). In right coset Ba , element a is referred to as representative of coset.

The map $aB \rightarrow (aB)' = Ba'$ map defines bijection between left cosets and B 's right cosets, so total of left cosets is equivalent to total of right cosets. The common value is called index of B in A .

Left cosets and right cosets are always the same in case of abelian groupings. Notation used switches to $a+B$ or $B+a$ if group operation is written additively.

Definition using Equivalence Classes :

Some authors define the left cosets of B in A as equivalence classes given by $x \sim y$ under equivalence relationship on A if and only if $x'y$ subset of B is given. Relation can also be described by $x \sim y$ if and only if $xb = y$ is described in B for certain b . It can be seen that given relation is simply an equivalence relationship and that two concepts are identical. Consequently, two left B -in- A cosets are either equivalent or disjoint. So, every element of A belongs to single left coset and so left cosets form partition of A . Similar claims for right cosets are also valid.

Double Cosets :

If A is group, B and C are subgroups of A , then in A double coset of B and C are sets of $BaC = \{bac : b \text{ an element of } B, c \text{ an element of } C\}$. These are left cosets of C and right cosets of B , respectively, if $B=1$ and $C=1$.

Notation :

Suppose A is group and B and C are subgroups of A .

- denotes set of left cosets of B in A .
- denotes set of right cosets of B in A .
- denotes set of double cosets of B and C in A .

Applications :

1. In computational group theory, cosets are essential.
2. Cosets play key role in the theorem for Lagrange.
3. The Thistlethwaite's algorithm used to solve Rubik's Cube is highly based on cosets.
4. linear error-correction in obtained decoded data is done using cosets.
5. They are used to construct Vitali sets, kind of non-measurable package.

Permutation groups

In mathematics, a permutation group is a group G whose elements are permutations of a given set M and whose group operation is the composition of permutations in G (which are thought of as bijective functions from the set M to itself). The group of all permutations of a set M is the symmetric group of M , often written as $\text{Sym}(M)$.^[1] The term permutation group thus means a subgroup of the symmetric group. If $M = \{1, 2, \dots, n\}$ then, $\text{Sym}(M)$, the symmetric group on n letters is usually denoted by S_n .

By Cayley's theorem, every group is isomorphic to some permutation group.

The way in which the elements of a permutation group permute the elements of the set is called its group action. Group actions have applications in the study of symmetries, combinatorics and many other branches of mathematics, physics and chemistry.

Homomorphism

Let G be a group. A subgroup H of G is said to be a normal subgroup of G if for all $h \in H$ and $x \in G$, $x h x^{-1} \in H$

If $x H x^{-1} = \{x h x^{-1} \mid h \in H\}$ then H is normal in G if and only if $x H x^{-1} \subseteq H, \forall x \in G$

Statement: If G is an abelian group, then every subgroup H of G is normal in G .

Proof: Let any $h \in H, x \in G$, then

$$x h x^{-1} = x (h x^{-1})$$

$$x h x^{-1} = (x x^{-1}) h$$

$$x h x^{-1} = e h$$

$$x h x^{-1} = h \in H$$

Hence H is normal subgroup of G .

Group Homomorphism:

A homomorphism is a mapping $f: G \rightarrow G'$ such that $f(xy) = f(x)f(y), \forall x, y \in G$. The mapping f preserves the group operation although the binary operations of the group G and G' are different. Above condition is called the homomorphism condition.

Kernel of Homomorphism: - The Kernel of a homomorphism f from a group G to a group G' with identity e' is the set $\{x \in G \mid f(x) = e'\}$

The kernel of f is denoted by $\text{Ker } f$.

If $f: G \rightarrow G'$ is a homomorphism of G into G' , then the image set of f is the range, denoted by $f(G)$, of the map f . Thus

$$\text{Im } (f) = f(G) = \{f(x) \in G' \mid x \in G\}$$

If $f(G) = G'$, then G' is called a homomorphic image of G .

Isomorphism:

Let $(G_1, *)$ and $(G_2, 0)$ be two algebraic system, where $*$ and 0 both are binary operations. The systems $(G_1, *)$ and $(G_2, 0)$ are said to be isomorphic if there exists an isomorphic mapping $f: G_1 \rightarrow G_2$

When two algebraic systems are isomorphic, the systems are structurally equivalent and one can be obtained from another by simply remaining the elements and operation.

Example: Let $(A_1, *)$ and (A_2, \square) be the two algebraic systems as shown in fig. Determine whether the two algebraic systems are isomorphic.

Solution: The two algebraic system $(A_1, *)$ and (A_2, \square) are isomorphic and (A_2, \square) is an isomorphic image of A_1 , such that

$$\begin{aligned} f(a) &= 1 \\ f(b) &= w \\ f(c) &= w^2 \end{aligned}$$

Automorphism:

Let $(G_1, *)$ and (G_2, \circ) be two algebraic system, where $*$ and \circ both are binary operations on G_1 and G_2 respectively. Then an isomorphism from $(G_1, *)$ to (G_2, \circ) is called an automorphism if $G_1 = G_2$

Rings:

An algebraic system $(R, +, \cdot)$ where R is a set with two arbitrary binary operations $+$ and \cdot , is called a ring if it satisfies the following conditions

1. $(R, +)$ is an abelian group.
2. (R, \cdot) is a semigroup.
3. The multiplication operation, is distributive over the addition operation i.e.,
 $a(b+c) = ab + ac$ and $(b+c)a = ba + ca$ for all $a, b, c \in R$.

Example1: Consider M be the set of all matrices of the type $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ over integers under matrix addition and matrix multiplication. Thus M form a ring.

Example2: The set $Z_9 = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ under the operation addition and multiplication modulo 9 forms a ring.

Types of Rings:

1. Commutative Rings: A ring $(R, +, \cdot)$ is called a commutative ring if it holds the commutative law under the operation of multiplication i.e., $a \cdot b = b \cdot a$, for every $a, b \in R$

Example1: Consider a set E of all even integers under the operation of addition and multiplication. The set E forms a commutative ring.

2. Ring with Unity: A ring $(R, +, \cdot)$ is called a ring with unity, if it has a multiplicative identity i.e.,

Example: Consider a set M of all 2×2 matrices over integers under matrix multiplication and matrix addition. The set M forms a ring with unity I .

3. Ring with Zero Divisions: If $a \cdot b = 0$, where a and b are any two non-zero elements of R in the ring $(R, +)$ then a and b are called divisions of zero and the ring $(R, +)$ is called ring with zero division.

4. Rings without Zero Division: An algebraic system $(R, +)$ where R is a set with two arbitrary binary operation $+$ and is called a ring without divisors of zero if for every $a, b \in R$, we have $a \cdot b \neq 0 \implies a \neq 0$ and $b \neq 0$

SubRings:

A subset A of a ring $(R, +)$ is called a subring of R , if it satisfies following conditions:

$(A, +)$ is a subgroup of the group $(R, +)$

A is closed under the multiplication operation i.e., $a \cdot b \in A$, for every $a, b \in A$.

Example: The ring $(\mathbb{I}, +)$ of integers is a subring of ring $(\mathbb{R}, +)$ of real numbers.

Isomorphism and Automorphism of groups

Isomorphism group

In abstract algebra, a group isomorphism is a function between two groups that sets up a one-to-one correspondence between the elements of the groups in a way that respects the given group operations. If there exists an isomorphism between two groups, then the groups are called isomorphic. From the standpoint of group theory, isomorphic groups have the same properties and need not be distinguished.

Definition and notation

Given two groups $(G, *)$ and $(H, \{\displaystyle \odot\})$, a group isomorphism from $(G, *)$ to $(H, \{\displaystyle \odot\})$ is a bijective group homomorphism from G to H . Spelled out, this means that a group isomorphism is a bijective function $\{\displaystyle f:G \rightarrow H\}$ such that for all u and v in G it holds that

$$\{\displaystyle f(u*v)=f(u)\odot f(v)\} \quad .$$

The two groups $(G, *)$ and $(H, \{\displaystyle \odot\})$ are isomorphic if there exists an isomorphism from one to the other. This is written:

$$\{(G, *) \cong (H, \cdot)\}$$

Often shorter and simpler notations can be used. When the relevant group operations are unambiguous they are omitted and one writes:

$$G \cong H$$

Sometimes one can even simply write $G = H$. Whether such a notation is possible without confusion or ambiguity depends on context. For example, the equals sign is not very suitable when the groups are both subgroups of the same group. See also the examples.

Conversely, given a group $(G, *)$, a set H , and a bijection $f: G \rightarrow H$, we can make H a group (H, \cdot) by defining

$$f(u) \cdot f(v) = f(u * v).$$

If $H = G$ and $\cdot = *$ then the bijection is an automorphism (q.v.).

Intuitively, group theorists view two isomorphic groups as follows: For every element g of a group G , there exists an element h of H such that h 'behaves in the same way' as g (operates with other elements of the group in the same way as g). For instance, if g generates G , then so does h . This implies in particular that G and H are in bijective correspondence. Thus, the definition of an isomorphism is quite natural.

An isomorphism of groups may equivalently be defined as an invertible morphism in the category of groups, where invertible here means has a two-sided inverse.

Automorphism group

In mathematics, the automorphism group of an object X is the group consisting of automorphisms of X . For example, if X is a finite-dimensional vector space, then the automorphism group of X is the general linear group of X , the group of invertible linear transformations from X to itself.

Especially in geometric contexts, an automorphism group is also called a symmetry group. A subgroup of an automorphism group is called a transformation group (especially in old literature).

- The automorphism group of a set X is precisely the symmetric group of X .
- A group homomorphism to the automorphism group of a set X amounts to a group action on X : indeed, each left G -action on a set X determines , and, conversely,

each homomorphism ρ defines an action by ρ .

- Let X and Y be two finite sets of the same cardinality and S_X the set of all bijections $X \rightarrow X$. Then S_X , which is a symmetric group (see above), acts on Y from the left freely and transitively; that is to say, Y is a torsor for S_X (cf. #In category theory).
- The automorphism group $\text{Aut}(C_n)$ of a finite cyclic group of order n is isomorphic to $S_{\phi(n)}$ with the isomorphism given by ϕ .^[1] In particular, $\text{Aut}(C_n)$ is an abelian group.
- Given a field extension L/K , its automorphism group is the group consisting of field automorphisms of L that fix K : it is better known as the Galois group of L/K .
- The automorphism group of the projective n -space over a field k is the projective linear group $\text{PGL}(n+1, k)$.^[2]
- The automorphism group of a finite-dimensional real Lie algebra \mathfrak{g} has the structure of a (real) Lie group (in fact, it is even a linear algebraic group: see below). If G is a Lie group with Lie algebra \mathfrak{g} , then the automorphism group of G has a structure of a Lie group induced from that on the automorphism group of \mathfrak{g} .^{[3][4]}
- Let P be a finitely generated projective module over a ring R . Then there is an embedding $\text{Aut}(P) \hookrightarrow \text{GL}(P)$, unique up to inner automorphisms.^[5]

UNIT –II

Propositional Logic:

Proposition

Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false. It is a technique of knowledge representation in logical and mathematical form.

Example:

1. a) It is Sunday.
2. b) The Sun rises from West (False proposition)
3. c) $3+3= 7$ (False proposition)
4. d) 5 is a prime number.

Following are some basic facts about propositional logic:

- Propositional logic is also called Boolean logic as it works on 0 and 1.
- In propositional logic, we use symbolic variables to represent the logic, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.
- Propositions can be either true or false, but it cannot be both.
- Propositional logic consists of an object, relations or function, and logical connectives.
- These connectives are also called logical operators.
- The propositions and connectives are the basic elements of the propositional logic.
- Connectives can be said as a logical operator which connects two sentences.
- A proposition formula which is always true is called tautology, and it is also called a valid sentence.

- A proposition formula which is always false is called Contradiction.
- A proposition formula which has both true and false values is called
- Statements which are questions, commands, or opinions are not propositions such as "Where is Rohini", "How are you", "What is your name", are not propositions.

Syntax of propositional logic:

The syntax of propositional logic defines the allowable sentences for the knowledge representation. There are two types of Propositions:

- a. Atomic Propositions
- b. Compound propositions
 - Atomic Proposition: Atomic propositions are the simple propositions. It consists of a single proposition symbol. These are the sentences which must be either true or false.

Example:

1. a) $2+2$ is 4, it is an atomic proposition as it is a true fact.
2. b) "The Sun is cold" is also a proposition as it is a false fact.
3. Compound proposition: Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives.

Example:

1. a) "It is raining today, and street is wet."
2. b) "Ankit is a doctor, and his clinic is in Mumbai."

Logical Connectives:

Logical connectives are used to connect two simpler propositions or representing a sentence logically. We can create compound propositions with the help of logical connectives. There are mainly five connectives, which are given as follows:

1. **Negation:** A sentence such as $\neg P$ is called negation of P. A literal can be either Positive literal or negative literal.
2. **Conjunction:** A sentence which has \wedge connective such as, **$P \wedge Q$** is called a conjunction.

Example: Rohan is intelligent and hardworking. It can be written as,

P= Rohan is intelligent,

Q= Rohan is hardworking. $\rightarrow P \wedge Q$.

3. **Disjunction:** A sentence which has \vee connective, such as **$P \vee Q$** . is called disjunction, where P and Q are the propositions.

Example: "Ritika is a doctor or Engineer",

Here P= Ritika is Doctor. Q= Ritika is Doctor, so we can write it as **$P \vee Q$** .

4. **Implication:** A sentence such as $P \rightarrow Q$, is called an implication. Implications are also known as if-then rules. It can be represented as
If it is raining, then the street is wet.
 Let P= It is raining, and Q= Street is wet, so it is represented as
 $P \rightarrow Q$

5. **Biconditional:** A sentence such as $P \Leftrightarrow Q$ is a Biconditional sentence, example If I am breathing, then I am alive
 P= I am breathing, Q= I am alive, it can be represented as P
 $\Leftrightarrow Q$.

Following is the summarized table for Propositional Logic Connectives:

Truth Table:

In propositional logic, we need to know the truth values of propositions in all possible scenarios. We can combine all the possible combination with logical connectives, and the representation of these combinations in a tabular format is called Truth table. Following are the truth table for all logical connectives:

First order logic

In the topic of Propositional logic, we have seen that how to represent statements using propositional logic. But unfortunately, in propositional logic, we can only represent the facts, which are either true or false. PL is not sufficient to represent the complex sentences or natural language statements. The propositional logic has very limited expressive power. Consider the following sentence, which we cannot represent using PL logic.

- **"Some humans are intelligent", or**
- **"Sachin likes cricket."**

To represent the above statements, PL logic is not sufficient, so we required some more powerful logic, such as first-order logic.

First-Order logic:

- First-order logic is another way of knowledge representation in artificial intelligence. It is an extension to propositional logic.
- FOL is sufficiently expressive to represent the natural language statements in a concise way.
- First-order logic is also known as **Predicate logic or First-order predicate logic**. First-order logic is a powerful language that develops information about the objects in a more easy way and can also express the relationship between those objects.
- First-order logic (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:
 - **Objects:** A, B, people, numbers, colors, wars, theories, squares, pits, wumpus,
 - **Relations: It can be unary relation such as:** red, round, is adjacent, **or n-any relation such as:** the sister of, brother of, has color, comes between
 - **Function:** Father of, best friend, third inning of, end of,

- As a natural language, first-order logic also has two main parts:
 - a. **Syntax**
 - b. **Semantics**

Syntax of First-Order logic:

The syntax of FOL determines which collection of symbols is a logical expression in first-order logic. The basic syntactic elements of first-order logic are symbols. We write statements in short-hand notation in FOL.

Basic Elements of First-order logic:

Following are the basic elements of FOL syntax:

Constant	1, 2, A, John, Mumbai, cat,....
Variables	x, y, z, a, b,....
Predicates	Brother, Father, >,....
Function	sqrt, LeftLegOf,
Connectives	$\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
Equality	$==$
Quantifier	\forall, \exists

Atomic sentences:

- Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- We can represent atomic sentences as **Predicate (term1, term2,, term n)**.

Example: Ravi and Ajay are brothers: \Rightarrow Brothers(Ravi, Ajay).
Chinky is a cat: \Rightarrow cat (Chinky).

Complex Sentences:

- Complex sentences are made by combining atomic sentences using connectives.

First-order logic statements can be divided into two parts:

- **Subject:** Subject is the main part of the statement.
- **Predicate:** A predicate can be defined as a relation, which binds two atoms together in a statement.

Consider the statement: "x is an integer.", it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.

Quantifiers in First-order logic:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:
 - a. **Universal Quantifier, (for all, everyone, everything)**
 - b. **Existential quantifier, (for some, at least one).**

Universal Quantifier:

Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.

The Universal quantifier is represented by a symbol \forall , which resembles an inverted A.

If x is a variable, then $\forall x$ is read as:

- **For all x**
- **For each x**
- **For every x.**

Example:

All man drink coffee.

Let a variable x which refers to a cat so all x can be represented in UOD as below:

$\forall x \text{ man}(x) \rightarrow \text{drink}(x, \text{coffee}).$

It will be read as: There are all x where x is a man who drink coffee.

Existential Quantifier:

Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.

It is denoted by the logical operator \exists , which resembles as inverted E. When it is used with a predicate variable then it is called as an existential quantifier.

If x is a variable, then existential quantifier will be $\exists x$ or $\exists(x)$. And it will be read as:

- **There exists a 'x.'**
- **For some 'x.'**
- **For at least one 'x.'**

Example:

Some boys are intelligent.

$\exists x: \text{boys}(x) \wedge \text{intelligent}(x)$

It will be read as: There are some x where x is a boy who is intelligent.

Points to remember:

- The main connective for universal quantifier \forall is implication \rightarrow .
- The main connective for existential quantifier \exists is and \wedge .

Properties of Quantifiers:

- In universal quantifier, $\forall x \forall y$ is similar to $\forall y \forall x$.
- In Existential quantifier, $\exists x \exists y$ is similar to $\exists y \exists x$.

- $\exists x \forall y$ is not similar to $\forall y \exists x$.

Some Examples of FOL using quantifier:

1. All birds fly.

In this question the predicate is "**fly(bird).**"

And since there are all birds who fly so it will be represented as follows.

$$\forall x \text{ bird}(x) \rightarrow \text{fly}(x).$$

2. Every man respects his parent.

In this question, the predicate is "**respect(x, y),**" where **x=man, and y= parent.**

Since there is every man so will use \forall , and it will be represented as follows:

$$\forall x \text{ man}(x) \rightarrow \text{respects}(x, \text{parent}).$$

3. Some boys play cricket.

In this question, the predicate is "**play(x, y),**" where **x= boys, and y= game.** Since there are some boys so we will use \exists , and it will be represented as:

$$\exists x \text{ boys}(x) \rightarrow \text{play}(x, \text{cricket}).$$

4. Not all students like both Mathematics and Science.

In this question, the predicate is "**like(x, y),**" where **x= student, and y= subject.**

Since there are not all students, so we will use \forall with negation, so following representation for this:

$$\neg \forall (x) [\text{student}(x) \rightarrow \text{like}(x, \text{Mathematics}) \wedge \text{like}(x, \text{Science})].$$

5. Only one student failed in Mathematics.

In this question, the predicate is "**failed(x, y),**" where **x= student, and y= subject.**

Since there is only one student who failed in Mathematics, so we will use following representation for this:

$$\exists (x) [\text{student}(x) \rightarrow \text{failed}(x, \text{Mathematics}) \wedge \forall (y) [\neg(x=y) \wedge \text{student}(y) \rightarrow \neg \text{failed}(y, \text{Mathematics})].$$

Free and Bound Variables:

The quantifiers interact with variables which appear in a suitable way. There are two types of variables in First-order logic which are given below:

Free Variable: A variable is said to be a free variable in a formula if it occurs outside the scope of the quantifier.

Example: $\forall x \exists (y)[P(x, y, z)],$ where **z is a free variable.**

Bound Variable: A variable is said to be a bound variable in a formula if it occurs within the scope of the quantifier.

Example: $\forall x [A(x) B(y)]$, here x and y are the bound variables.

Basic logical operations

1. Negation: It means the opposite of the original statement. If p is a statement, then the negation of p is denoted by $\sim p$ and read as 'it is not the case that p .' So, if p is true then $\sim p$ is false and vice versa.

Example: If statement p is Paris is in France, then $\sim p$ is 'Paris is not in France'.

p	$\sim p$
T	F
F	T

2. Conjunction: It means Anding of two statements. If p, q are two statements, then " p and q " is a compound statement, denoted by $p \wedge q$ and referred as the conjunction of p and q . The conjunction of p and q is true only when both p and q are true. Otherwise, it is false.

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

3. Disjunction: It means Oring of two statements. If p, q are two statements, then " p or q " is a compound statement, denoted by $p \vee q$ and referred to as the disjunction of p and q . The disjunction of p and q is true whenever at least one of the two statements is true, and it is false only when both p and q are false.

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

4. Implication / if-then (\rightarrow): An implication $p \rightarrow q$ is the proposition "if p, then q." It is false if p is true and q is false. The rest cases are true.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	F

5. If and Only If (\leftrightarrow): $p \leftrightarrow q$ is bi-conditional logical connective which is true when p and q are same, i.e., both are false or both are true.

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Derived Connectors

1. NAND: It means negation after ANDing of two statements. Assume p and q be two propositions. Nanding of p and q to be a proposition which is false when both p and q are true, otherwise true. It is denoted by $p \uparrow q$.

p	q	$p \uparrow q$
T	T	F
T	F	T
F	T	T
F	F	T

2. NOR or Joint Denial: It means negation after ORing of two statements. Assume p and q be two propositions. NORing of p and q to be a proposition which is true when both p and q are false, otherwise false. It is denoted by $p \downarrow q$.

p	q	$p \downarrow q$
T	T	F
T	F	F
F	T	F
F	F	T

3. XOR: Assume p and q be two propositions. XORing of p and q is true if p is true or q is true but not both and vice-versa. It is denoted by $p \oplus q$.

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T

F	F	F
---	---	---

Example1: Prove that $X \oplus Y \cong (X \wedge \sim Y) \vee (\sim X \wedge Y)$.

Solution: Construct the truth table for both the propositions.

X	Y	$X \oplus Y$	$\sim Y$	$\sim X$	$X \wedge \sim Y$	$\sim X \wedge Y$	$(X \wedge \sim Y) \vee (\sim X \wedge Y)$
T	T	F	F	F	F	F	F
T	F	T	T	F	T	F	T
F	T	T	F	T	F	T	T
F	F	F	T	T	F	F	F

As the truth table for both the proposition is the same.

1. $X \oplus Y \cong (X \wedge \sim Y) \vee (\sim X \wedge Y)$. Hence Proved.

Example2: Show that $(p \oplus q) \vee (p \downarrow q)$ is equivalent to $p \uparrow q$.

Solution: Construct the truth table for both the propositions.

p	q	$p \oplus q$	$(p \downarrow q)$	$(p \oplus q) \vee (p \downarrow q)$	$p \uparrow q$
T	T	F	F	F	F
T	F	T	F	T	T
F	T	T	F	T	T
F	F	F	T	T	T

Tautologies

A proposition P is a tautology if it is true under all circumstances. It means it contains the only T in the final column of its truth table.

Example: Prove that the statement $(p \rightarrow q) \leftrightarrow (\sim q \rightarrow \sim p)$ is a tautology.

Solution: Make the truth table of the above statement:

p	q	$p \rightarrow q$	$\sim q$	$\sim p$	$\sim q \rightarrow \sim p$	$(p \rightarrow q) \leftrightarrow (\sim q \rightarrow \sim p)$
T	T	T	F	F	T	T
T	F	F	T	F	F	T
F	T	T	F	T	T	T
F	F	T	T	T	T	T

As the final column contains all T's, so it is a tautology.

Contradiction:

A statement that is always false is known as a contradiction.

Example: Show that the statement $p \wedge \sim p$ is a contradiction.

Solution:

p	$\sim p$	$p \wedge \sim p$
T	F	F
F	T	F

Since, the last column contains all F's, so it's a contradiction.

Contingency:

A statement that can be either true or false depending on the truth values of its variables is called a contingency.

p	q	$p \rightarrow q$	$p \wedge q$	$(p \rightarrow q) \rightarrow (p \wedge q)$
T	T	T	T	T
T	F	F	F	T

F	T	T	F	F
F	F	T	F	F

Contradictions

A proposition P is a tautology if it is true under all circumstances. It means it contains the only T in the final column of its truth table.

Example: Prove that the statement $(p \rightarrow q) \leftrightarrow (\sim q \rightarrow \sim p)$ is a tautology.

Solution: Make the truth table of the above statement:

p	q	$p \rightarrow q$	$\sim q$	$\sim p$	$\sim q \rightarrow \sim p$	$(p \rightarrow q) \leftrightarrow (\sim q \rightarrow \sim p)$
T	T	T	F	F	T	T
T	F	F	T	F	F	T
F	T	T	F	T	T	T
F	F	T	T	T	T	T

As the final column contains all T's, so it is a tautology.

Contradiction:

A statement that is always false is known as a contradiction.

Example: Show that the statement $p \wedge \sim p$ is a contradiction.

Solution:

p	$\sim p$	$p \wedge \sim p$
T	F	F
F	T	F

Since, the last column contains all F's, so it's a contradiction.

Contingency:

A statement that can be either true or false depending on the truth values of its variables is called a contingency.

p	q	$p \rightarrow q$	$p \wedge q$	$(p \rightarrow q) \rightarrow (p \wedge q)$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	F
F	F	T	F	F

Algebra of Proposition

The rules of mathematical logic specify methods of reasoning mathematical statements. Greek philosopher, Aristotle, was the pioneer of logical reasoning. Logical reasoning provides the theoretical base for many areas of mathematics and consequently computer science. It has many practical applications in computer science like design of computing machines, artificial intelligence, definition of data structures for programming languages etc.

Propositional Logic is concerned with statements to which the truth values, "true" and "false", can be assigned. The purpose is to analyze these statements either individually or in a composite manner.

Propositional Logic – Definition

A proposition is a collection of declarative statements that has either a truth value "true" or a truth value "false". A propositional consists of propositional variables and connectives. We denote the propositional variables by capital letters (A, B, etc). The connectives connect the propositional variables.

Some examples of Propositions are given below –

- "Man is Mortal", it returns truth value "TRUE"
- " $12 + 9 = 3 - 2$ ", it returns truth value "FALSE"

The following is not a Proposition –

- "A is less than 2". It is because unless we give a specific value of A, we cannot say whether the statement is true or false.

Connectives

In propositional logic generally we use five connectives which are –

- OR (\vee)
- AND (\wedge)
- Negation/ NOT (\neg)
- Implication / if-then (\rightarrow)
- If and only if (\Leftrightarrow).

OR (\vee) – The OR operation of two propositions A and B (written as $A \vee B$) is true if at least any of the propositional variable A or B is true.

The truth table is as follows –

A	B	$A \vee B$
True	True	True
True	False	True
False	True	True
False	False	False

AND (\wedge) – The AND operation of two propositions A and B (written as $A \wedge B$) is true if both the propositional variable A and B is true.

The truth table is as follows –

A	B	$A \wedge B$
True	True	True
True	False	False
False	True	False

False	False	False
-------	-------	-------

Negation (\neg) – The negation of a proposition A (written as $\neg A$) is false when A is true and is true when A is false.

The truth table is as follows –

A	$\neg A$
True	False
False	True

Implication / if-then (\rightarrow) – An implication $A \rightarrow B$ is the proposition “if A, then B”. It is false if A is true and B is false. The rest cases are true.

The truth table is as follows –

A	B	$A \rightarrow B$
True	True	True
True	False	False
False	True	True
False	False	True

If and only if (\Leftrightarrow) – $A \Leftrightarrow B$ is bi-conditional logical connective which is true when p and q are same, i.e. both are false or both are true.

The truth table is as follows –

A	B	$A \leftrightarrow B$
True	True	True
True	False	False
False	True	False
False	False	True

Tautologies

A Tautology is a formula which is always true for every value of its propositional variables.

Example – Prove $[(A \rightarrow B) \wedge A] \rightarrow B$ is a tautology

The truth table is as follows –

A	B	$A \rightarrow B$	$(A \rightarrow B) \wedge A$	$[(A \rightarrow B) \wedge A] \rightarrow B$
True	True	True	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	True

As we can see every value of $[(A \rightarrow B) \wedge A] \rightarrow B$ is "True", it is a tautology.

Contradictions

A Contradiction is a formula which is always false for every value of its propositional variables.

Example – Prove $(A \vee B) \wedge (\neg A) \wedge (\neg B)$ is a contradiction

The truth table is as follows –

A	B	$A \vee B$	$\neg A$	$\neg B$	$(\neg A) \wedge (\neg B)$	$(A \vee B) \wedge [(\neg A) \wedge (\neg B)]$
True	True	True	False	False	False	False
True	False	True	False	True	False	False
False	True	True	True	False	False	False
False	False	False	True	True	True	False

As we can see every value of $(A \vee B) \wedge (\neg A) \wedge (\neg B)$ is “False”, it is a contradiction.

Contingency

A Contingency is a formula which has both some true and some false values for every value of its propositional variables.

Example – Prove $(A \vee B) \wedge (\neg A)$ a contingency

The truth table is as follows –

A	B	$A \vee B$	$\neg A$	$(A \vee B) \wedge (\neg A)$
True	True	True	False	False
True	False	True	False	False
False	True	True	True	True

False	False	False	True	False
-------	-------	-------	------	-------

As we can see every value of $(A \vee B) \wedge (\neg A)$ has both "True" and "False", it is a contingency.

Propositional Equivalences

Two statements X and Y are logically equivalent if any of the following two conditions hold –

- The truth tables of each statement have the same truth values.
- The bi-conditional statement $X \Leftrightarrow Y$ is a tautology.

Example – Prove $\neg(A \vee B)$ and $(\neg A) \wedge (\neg B)$ are equivalent

Testing by 1st method (Matching truth table)

A	B	$A \vee B$	$\neg(A \vee B)$	$\neg A$	$\neg B$	$(\neg A) \wedge (\neg B)$
True	True	True	False	False	False	False
True	False	True	False	False	True	False
False	True	True	False	True	False	False
False	False	False	True	True	True	True

Here, we can see the truth values of $\neg(A \vee B)$ and $(\neg A) \wedge (\neg B)$ are same, hence the statements are equivalent.

Testing by 2nd method (Bi-conditionality)

A	B	$\neg (A \vee B)$	$[(\neg A) \wedge (\neg B)]$	$[\neg (A \vee B)] \Leftrightarrow [(\neg A) \wedge (\neg B)]$
True	True	False	False	True
True	False	False	False	True
False	True	False	False	True
False	False	True	True	True

As $[\neg(A \vee B)] \Leftrightarrow [(\neg A) \wedge (\neg B)]$ is a tautology, the statements are equivalent.

Inverse, Converse, and Contra-positive

Implication / if-then (\rightarrow) is also called a conditional statement. It has two parts –

- Hypothesis, p
- Conclusion, q

As mentioned earlier, it is denoted as $p \rightarrow q$.

Example of Conditional Statement – “If you do your homework, you will not be punished.” Here, “you do your homework” is the hypothesis, p, and “you will not be punished” is the conclusion, q.

Inverse – An inverse of the conditional statement is the negation of both the hypothesis and the conclusion. If the statement is “If p, then q”, the inverse will be “If not p, then not q”. Thus the inverse of $p \rightarrow q$ is $\neg p \rightarrow \neg q$.

Example – The inverse of “If you do your homework, you will not be punished” is “If you do not do your homework, you will be punished.”

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 $q \rightarrow p$.

Example – The converse of “If you do your homework, you will not be punished” is “If you will not be punished, you do your homework”.

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 $\neg q \rightarrow \neg p$.

Example – The Contra-positive of " If you do your homework, you will not be punished" is "If you are punished, you did not do your homework".

Duality Principle

Duality principle states that for any true statement, the dual statement obtained by interchanging unions into intersections (and vice versa) and interchanging Universal set into Null set (and vice versa) is also true. If dual of any statement is the statement itself, it is said **self-dual** statement.

Example – The dual of $(A \cap B) \cup C$ is $(A \cup B) \cap C$.

Normal Forms

We can convert any proposition in two normal forms –

- Conjunctive normal form
- Disjunctive normal form

Conjunctive Normal Form

A compound statement is in conjunctive normal form if it is obtained by operating AND among variables (negation of variables included) connected with ORs. In terms of set operations, it is a compound statement obtained by Intersection among variables connected with Unions.

Examples

- $(A \vee B) \wedge (A \vee C) \wedge (B \vee C \vee D)$
- $(P \cup Q) \cap (Q \cup R)$

Disjunctive Normal Form

A compound statement is in disjunctive normal form if it is obtained by operating OR among variables (negation of variables included) connected with ANDs. In terms of set operations, it is a compound statement obtained by Union among variables connected with Intersections.

Examples

- $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C \wedge D)$
- $(P \cap Q) \cup (Q \cap R)$

Logical implication

A Logical Connective is a symbol which is used to connect two or more propositional or predicate logics in such a manner that resultant logic depends only on the input logics and the meaning of the connective used.

Generally there are five connectives which are –

- OR (\vee)
- AND (\wedge)
- Negation/ NOT (\neg)
- Implication / if-then (\rightarrow)
- If and only if (\Leftrightarrow).

OR (\vee) – The OR operation of two propositions A and B (written as $A \vee B$) is true if at least any of the propositional variable A or B is true.

The truth table is as follows –

A	B	$A \vee B$
True	True	True
True	False	True
False	True	True
False	False	False

AND (\wedge) – The AND operation of two propositions A and B (written as $A \wedge B$) is true if both the propositional variable A and B is true.

The truth table is as follows –

A	B	$A \wedge B$
True	True	True
True	False	False

A	B	$A \wedge B$
False	True	False
False	False	False

Negation (\neg) – The negation of a proposition A (written as $\neg A$) is false when A is true and is true when A is false.

The truth table is as follows –

A	$\neg A$
True	False
False	True

Implication / if-then (\rightarrow) – An implication $A \rightarrow B$ is the proposition “if A, then B”. It is false if A is true and B is false. The rest cases are true.

The truth table is as follows –

A	B	$A \rightarrow B$
True	True	True
True	False	False
False	True	True
False	False	True

If and only if (\Leftrightarrow) – $A \Leftrightarrow B$ is bi-conditional logical connective which is true when p and q are same, i.e. both are false or both are true.

The truth table is as follows –

A	B	$A \Leftrightarrow B$
True	True	True
True	False	False
False	True	False
False	False	True

Logical equivalence

In this lesson, we have discussed the basis of propositional logic. With this logic, students can find the truth and falsity of the statements.

After reading this lesson, you should be able to understand

- What is propositional logic? and its usage in reasoning.
- Types of statements
- Connectives for forming compound statements

7.1 Introduction

Logic, logical thinking, and correct reasoning have wide applications in many fields, including law, psychology, rhetoric, science, and mathematics. While an interesting study can be made of logic in human lives, we shall restrict our attention mainly to logic as it is used in mathematics. This logic was first studied systematically by Aristotle (384 B.C.-322 B.C.). Aristotle and his followers studied patterns of correct and incorrect reasoning.

Medieval philosophers and theologians, who made an intimate study of logical arguments, carried the work of Aristotle forward. A big advance in the study of mathematical logic came with the work of Gottfried Wilhelm von Leibniz (1646-1716), one of the inventors of calculus. Leibniz introduced symbols to represent ideas in logic- letters for statements and other symbols for the relations between statements. Leibniz hoped that logic would become a universal characteristic and unify all of mathematics.

Logic is the tool for reasoning about the truth and falsity of statements. There are two main directions in which logic develops.

- The first is the depth to which we explore the structure of statements. The study of the basic level of structure is called propositional logic. First order predicate logic, which is often called just predicate logic, studies structure on a deeper level.
- The second direction is the nature of truth. For example, one may talk about statements that are usually true or true at certain times.

“True” and “false” could be replaced by T and F (or any other two symbols) in our discussions. Using T and F relates logic to Boolean functions. In fact, propositional logic is the study of Boolean functions, where T plays the role of “true” and F the role of “false.”

Our study is restricted to propositional and predicate logic only.

7.2 Propositional calculus

7.2.1 Propositions

A declarative sentence that is either true or false, but not both, is called a proposition.

In mathematics, the propositions are denoted by alphabets known as propositional variables. The conventionally used alphabets are p, q, r, s, and so on. The truth-value of a proposition is true, denoted by T, if it is a true proposition and false, denoted by F, if it is a false proposition. Here the letters T and F are constants.

Example for a true proposition:

Chennai is the state capital of Tamil Nadu. — (1)

The truth-value of the above statement is true, denoted by T.

Example for a false proposition:

Oxygen is a solid. — (2)

The truth-value of the above statement is false, denoted by F.

The area of logic that deals with propositions is called the propositional calculus or propositional logic.

7.2.2 Types of Propositions

There are two types of propositions namely

1) Simple proposition and

2) Compound proposition.

A simple proposition is one in which the sentences cannot be further broken into simple or atomic sentences. Two or more simple propositions connected by operators is known as a compound proposition. The operators are known as logical connectives or simply connectives. The logical connectives are shown in the table 3.1.

Table 3.1 – Logical connectives.

Symbol	Connective	Type of statement
\emptyset	not	Negation
$\dot{\cup}$	and	Conjunction
$\dot{\cup}$	or	Disjunction
$\textcircled{\text{R}}$	implies	implication or conditional
\ll	if and only if	equivalence or biconditional

Truth Table: It shows the relationship between the truth-value of a compound proposition and the truth-values of its constituent simple propositions.

7.2.3 Basic Logical Operations

The basic logical operations are conjunction, disjunction and negation.

Conjunction

Let p and q be propositions. The proposition “ p and q ”, denoted by $p \dot{\cup} q$, is true when both p and q are true and is false otherwise. The statement $p \dot{\cup} q$ is called the conjunction of p and q .

The truth table for the conjunctions of two propositions is given in the table 3.2.

Table 3.2

p	q	$p \dot{\cup} q$
T	T	T
T	F	F
F	T	F
F	F	F

Illustration

a) Consider the following four compound statements

- 1) Bharathiar University is at Coimbatore and $2+2 = 4$.
- 2) Bharathiar University is at Coimbatore and $2+2 = 5$.
- 3) Bharathiar University is at Chennai and $2+2 = 4$.
- 4) Bharathiar University is at Chennai and $2+2 = 5$.

Only the first statement is true. Each of the other statements is false, since at least one of its simple statements is false.

Disjunction

Let p and q be propositions. The proposition “ p or q ”, denoted by $p \cup q$, is false when both p and q are false and is true otherwise. The statement $p \cup q$ is called the disjunction of p and q .

The truth table for the disjunctions of two propositions is given in the table 3.3.

Table 3.3

p	q	$p \cup q$
T	T	T
T	F	T
F	T	T
F	F	F

Illustration

b) Consider the following four compound statements

- 1) Bharathiar University is at Coimbatore or $2+2 = 4$.
- 2) Bharathiar University is at Coimbatore or $2+2 = 5$.
- 3) Bharathiar University is at Chennai or $2+2 = 4$.
- 4) Bharathiar University is at Chennai or $2+2 = 5$.

Only the last statement is false. Each of the other statements is true, since at least one of its simple statements is true.

The disjunction may be either Inclusive or Exclusive. In the inclusive disjunction the compound statement $p \cup q$ is true only when at least one of the statement is true. But in the exclusive disjunction, commonly called inequivalence, the compound statement p

$\cup q$ is true only when either p or q but not both, is true.

Example 1

Let p : ABC Company earned 20% profit per share in 2005. q : ABC Company paid 12% dividend per share in 2005.

The inclusive disjunction of p and q is

$p \cup q$: ABC Company earned 20% profit per share in 2005 or ABC Company paid 12% dividend per share in 2005 or both.

The exclusive disjunction of p and q is

$p \oplus q$: ABC Company earned 20% profit per share in 2005 or ABC Company paid 12% dividend per share in 2005 but not both.

Negation

The negation of a true statement is false, and the negation of a false statement is true. Its truth table is given in the table 3.4.

Table 3.4

p	$\neg p$
T	F
F	T

7.2.4 Derived Connectives

NAND

It is obtained by negating the result of ANDing of two statements.

For example, If p and q are two statements then NANDing of these two statements, denoted by $p \downarrow q$, is false when both p and q are true, otherwise true. Its truth table is given in the table 3.5.

Table 3.5

p	q	$p \downarrow q$
T	T	F
T	F	T
F	T	T
F	F	T

NOR

It is obtained by negating the result of ORing of two statements.

For example, if p and q are two statements then ORing of these two statements, denoted by $p \uparrow q$, is true when both p and q are false, otherwise false. Its truth table is given in the table 3.6.

Table 3.6

p	q	$p \uparrow q$
---	---	----------------

T	T	F
T	F	F
F	T	F
F	F	T

XOR

If p and q are two statements then XORing of these two statements, denoted by p

$\dot{\wedge}$ q, is false when both p and q are same, otherwise true. Its truth table is given in the table 3.7.

Table 3.7

p	q	$p \dot{\wedge} q$
T	T	F
T	F	T
F	T	T
F	F	F

7.3 Conditional statement

A conditional statement is a compound statement that uses the connective if...then. For example, the statement

If I read for too long, then I get a headache.

In the above statement, the component coming after the word if gives a condition (but not necessarily the only condition) under which the statement coming after then will be true.

The conditional is written with an arrow, so that “if p, then q” is symbolized as $p \rightarrow q$. We read $p \rightarrow q$ as “p implies q” or “if p, then q”. In the conditional $p \rightarrow q$, the statement p is the antecedent, while q is the consequent.

The conditional connective may not always be explicitly stated. That is, it may be “hidden” in an everyday expression. For example, the statement

“It is difficult to study when you are distracted”

can be written

“If you are distracted then it is difficult to study”.

Truth table for the conditional “if p, then q” is given in 3.8

Table 3.8

p q p \Rightarrow q

T T T

T F F

F T T

F F T

7.3.1 Special characteristics of conditional statements

1) p \Rightarrow q is false only when the antecedent is true and the consequent is false.

2) If the antecedent is false, then p \Rightarrow q is automatically true.

3) If the consequent is true, then p \Rightarrow q is automatically true.

Negation of p \Rightarrow q

The negation of p \Rightarrow q is p \wedge \neg q.

Conditional as a disjunction

A conditional may be written as a disjunction as below.

p \Rightarrow q is equivalent to \neg p \vee q.

7.3.2 Converse, Inverse and Contrapositive

An conditional statement is made up of an antecedent and a consequent. If they are interchanged, negated or both, a new conditional statement is formed. Suppose that we begin with the direct statement

“If you stay, then I go,”

and interchange the antecedent (“you stay”) and the consequent (“I go”). We obtain the new conditional statement

“If I go, then you stay.”

This new conditional is called the converse of the given statement.

By negating both the antecedent and the consequent, we obtain the inverse of the given statement.

“If you do not stay, then I do not go.”

If the antecedent and the consequent are both interchanged and negated, the contrapositive of the given statement is formed:

“If I do not go, then you do not stay.”

These three related statements for the conditional $p \rightarrow q$ are summarized below.

Related conditional Statements		
Direct statement	$p \rightarrow q$	(If p , then q .)
Converse	$q \rightarrow p$	(If q , then p .)
Inverse	$\neg p \rightarrow \neg q$	(If not p , then not q .)
Contrapositive	$\neg q \rightarrow \neg p$	(If not q , then not p .)

7.4 Biconditional statement

In elementary algebra we learn that both of these statements are true:

If $x > 0$, then $5x > 0$.

If $5x > 0$, then $x > 0$.

Notice that the second statement is the converse of the first. If we wish to make the statement that each condition ($x > 0$, $5x > 0$) implies the other, we use the following language:

$x > 0$ if and only if $5x > 0$. This also may be stated as

$5x > 0$ if and only if $x > 0$.

The compound statement p if and only if q is called a biconditional. It is symbolized $p \leftrightarrow q$, and is interpreted as the conjunction of the two conditionals $p \rightarrow q$ and $q \rightarrow p$. Using symbols, this conjunction is written

$$(p \rightarrow q) \wedge (q \rightarrow p)$$

so that, by definition,

$$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p).$$

Using this definition, the truth table for the biconditional $p \leftrightarrow q$ can be determined as shown in table 3.9.

Table 3.9

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Example 2

Tell whether each biconditional statement is true or false.

(a) $6+9=15$ if and only if $12+4=16$

Both $6 + 9 = 15$ and $12 + 4 = 16$ are true. By the truth table for the biconditional, this biconditional is true.

(b) $5 + 2 = 10$ if and only if $17 + 19 = 36$.

Since the first component ($5 + 2 = 10$) is false, and the second is true, the entire biconditional statement is false.

(c) $6=5$ if and only if $12=12$

Both component statements are false, so by the last line of the truth table for the biconditional, the entire statement is true. (Understanding this might take some extra thought!) .

7.5 Problems and Solutions

1) Make truth tables for

(a) $(p \rightarrow q) \leftrightarrow (p \rightarrow r)$ (b) $p \rightarrow q \rightarrow r$ (c) $p \leftrightarrow q \leftrightarrow r$

Solution (a) Truth table is shown in Table 3.10

Truth table 3.10: $(p \rightarrow q) \leftrightarrow (p \rightarrow r)$

p	Q	r	$(p \rightarrow q)$	$(p \rightarrow r)$	$(p \rightarrow q) \leftrightarrow (p \rightarrow r)$
T	T	T	F	F	F
T	T	F	F	F	F
T	F	T	F	F	F
T	F	F	F	F	F

F	T	T	F	F	F
F	T	F	F	T	F
F	F	T	T	F	F
F	F	T	T	T	T

(b) The truth table is shown in Table 3.11

Truth table: 3.11 $p - q - r$

p	Q	r	$p - q$	$p - q - r$
T	T	T	F	T
T	T	F	F	T
T	F	T	T	F
T	F	F	T	T
F	T	T	T	F
F	T	F	T	T
F	F	T	T	F
F	F	F	T	T

(c) The truth table is shown in Table 3.12.

Truth table: 3.12 $p \dot{\wedge} q \dot{\wedge} r$

p	q	r	$p \dot{\wedge} q$	$p \dot{\wedge} q \dot{\wedge} r$
T	T	T	F	T
T	T	F	F	F
T	F	T	T	F
T	F	F	T	T
F	T	T	T	F
F	T	F	T	T
F	F	T	F	T
F	F	F	F	F

2) If p and q are two statements, then show that the statement $(p - q) \dot{\wedge} (p - q)$ is equivalent to $(p \dot{\cup} q) \dot{\cup} (p^{-} q)$.

Solution: The equivalence of two compound statements is shown in the truth Table 3.13.

Truth table: 3.13 $(p - q) \dot{\wedge} (p - q)$ and $(p \dot{\cup} q) \dot{\cup} (p^{-} q)$

p	Q	$p - q$	$(p - q) \dot{\wedge} (p - q)$	$(p \dot{\cup} q)$	$(p^{-} q)$	$(p \dot{\cup} q) \dot{\cup} (p^{-} q)$
(1)	(2)	(3)	(4)	(5)	(6)	(7)
T	T	F	F	T	F	F
T	F	T	F	T	F	F
F	T	T	F	T	F	F
F	F	T	F	F	T	F

Since values in Columns (4) and (7) are same, therefore two statements are equivalent.

3) If p and q are two statements, then show that $p \dot{\wedge} q$ is equivalent to $(p \dot{\cup} \emptyset q) \dot{\cup} (\emptyset p \dot{\cup} g)$.

Solution: The equivalence of two compound statements is shown in truth Table 3.14.

Truth table 3.14 : $p \dot{\wedge} q$ and $(p \dot{\cup} \emptyset q) \dot{\cup} (\emptyset p \dot{\cup} g)$

p	q	$p \dot{\wedge} q$	$\emptyset p$	$\emptyset q$	$p \dot{\cup} \emptyset q$	$\emptyset p \dot{\cup} g$	$(p \dot{\cup} \emptyset q) \dot{\cup} (\emptyset p \dot{\cup} g)$
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
T	T	F	F	F	F	F	F
T	F	T	T	T	T	F	T
F	T	T	F	F	F	T	T
F	F	F	T	T	F	F	F

Since values in Columns (3) and (8) are same, therefore two compound statements are equivalent.

4) If p and q are two statements, then show that the statement $(p \dot{\wedge} q) \dot{\cup} (p^{-} q)$ is equivalent to $p - q$.

Solution: The equivalence of two compound statements is shown in truth Table 3.15.

Truth table: 3.15 $(p \dot{\wedge} q) \dot{\cup} (p^{-} q)$ and $p - q$

p	q	$(p \dot{\wedge} q)$	$(p^{-} q)$	$(p \dot{\wedge} q) \dot{\cup} (p^{-} q)$	$p - q$
(1)	(2)	(3)	(4)	(5)	(6)
T	T	F	F	F	F

T	F	T	F	T	T
F	T	T	F	T	T
F	F	F	T	T	T

Since values in Columns (5) and (6) are same, therefore two compound statements are equivalent.

7.6 Let Us Sum Up

Symbols Used in this Chapter

Connectives	Symbols	Types of Statements
and	$\dot{\cup}$	Conjunction
or	$\dot{\cup}$	Disjunction
not	\emptyset	Negation
if. . . then	$\textcircled{\text{R}}$	Conditional
if and only if	\ll	Biconditional

Truth Tables

$p \ \emptyset \ p$
T F
F T

p	q	$p \dot{\cup} q$	$p \dot{\cup} q$	$p - q$	$p^- q$	$p \dot{\Delta} q$	$p \textcircled{\text{R}} q$	$p \ll q$
T	T	T	T	F	F	F	T	T
T	F	F	T	T	F	T	F	F
F	T	F	T	T	F	T	T	F
F	F	F	F	T	T	F	T	T

Statements Related to Conditional

Direct statement $p \textcircled{\text{R}} q$ (If p, then q.)

Converse	$q \Rightarrow p$	(If q, then p.)
Inverse	$\neg p \Rightarrow \neg q$	(If not p, then not q.)
Contrapositive	$\neg q \Rightarrow \neg p$	(If not q, then not p.)

7.7 Lesson End Activities

Write a negation for each of the following statements.

- $1.5+3=9$
- Every good boy deserves favour.
- Some people here can't play this game.
- If it ever comes to that, I won't be here.
- My mind is made up and you can't change it.

Let p represent "it is broken" and let q represent "you can fix it." Write each of the following in symbols.

- If it isn't broken, then you can fix it.
- It is broken or you can't fix it.
- You can't fix anything that is broken.

Using the same directions as for Exercises 6-8, write each of the following in words.

9. $\neg p \Rightarrow q$ 10. $p \Leftrightarrow \neg q$

In each of the following, assume that p and q are true, with r false. Find the truth value of

each statement.

- $\neg p \Rightarrow \neg r$
- $r \Rightarrow (p \Rightarrow q)$
- $r \Rightarrow (s \Rightarrow r)$ (The truth value of the statement s is unknown.)
- $r \Leftrightarrow (p \Rightarrow \neg q)$

15. What are the necessary conditions for a conditional statement to be false? for a conjunction to be true?

16. Explain in your own words why, if p is a statement, the biconditional $p \Leftrightarrow \neg p$ must be false.

Write a truth table for each of the following. Identify any tautologies.

17. $p \cup (\neg p \cup q)$ 18. $\neg(p \cup q) \leftrightarrow (\neg p \cup \neg q)$
 Decide whether each statement is true or false.

19. All positive integers are whole numbers.

20. If $x + 4 = 6$, then $x > 1$.

Write each conditional statement in the form if . . . then.

21. All rational numbers are real numbers.

22. Being a rectangle is sufficient for a polygon to be a quadrilateral.

23. Being divisible by 2 is necessary for a number to be divisible by 6.

24. She cries only if she is hurt.

For each statement, write (a) the converse, (b) the inverse, and (c) the contrapositive.

25. If a picture paints a thousand words, the graph will help me understand it.

26. $\neg p \leftrightarrow (q \cup r)$ (Use one of De Morgan's laws as necessary.)

Normal forms

The problem of finding whether a given statement is tautology or contradiction or satisfiable in a finite number of steps is called the Decision Problem. For Decision Problem, construction of truth table may not be practical always. We consider an alternate procedure known as the reduction to normal forms.

There are two such forms:

1. Disjunctive Normal Form (DNF)
2. Conjunctive Normal Form

Disjunctive Normal Form (DNF): If p, q are two statements, then " p or q " is a compound statement, denoted by $p \vee q$ and referred as the disjunction of p and q . The disjunction of p and q is true whenever at least one of the two statements is true, and it is false only when both p and q are false

p	q	$p \vee q$
T	T	T
T	F	T

F	T	T
F	F	F

Example: - if p is "4 is a positive integer" and q is " $\sqrt{5}$ is a rational number", then $p \vee q$ is true as statement p is true, although statement q is false.

Conjunctive Normal Form: If p, q are two statements, then "p and q" is a compound statement, denoted by $p \wedge q$ and referred as the conjunction of p and q. The conjunction of p and q is true only when both p and q are true, otherwise, it is false

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Example: if statement p is " $6 < 7$ " and statement q is " $-3 > -4$ " then the conjunction of p and q is true as both p and q are true statements.

Inference Theory

To deduce new statements from the statements whose truth that we already know, Rules of Inference are used.

What are Rules of Inference for?

Mathematical logic is often used for logical proofs. Proofs are valid arguments that determine the truth values of mathematical statements.

An argument is a sequence of statements. The last statement is the conclusion and all its preceding statements are called premises (or hypothesis). The symbol " \therefore ", (read therefore) is placed before the conclusion. A valid argument is one where the conclusion follows from the truth values of the premises.

Rules of Inference provide the templates or guidelines for constructing valid arguments from the statements that we already have.

Table of Rules of Inference

Rule of Inference	Name	Rule of Inference	Name
$P \therefore P \vee Q$ $Q \therefore P \vee Q$	Addition	$P \vee Q \neg P \therefore Q$ $P \vee Q \neg Q \therefore P$	Disjunctive Syllogism
$PQ \therefore P \wedge Q$ $PQ \therefore P \wedge Q$	Conjunction	$P \rightarrow Q, Q \rightarrow R \therefore P \rightarrow R$ $P \rightarrow Q, Q \rightarrow R \therefore P \rightarrow R$	Hypothetical Syllogism
$P \wedge Q \therefore P$ $P \wedge Q \therefore Q$	Simplification	$(P \rightarrow Q) \wedge (R \rightarrow S), P \vee R \therefore Q \vee S$ $(P \rightarrow Q) \wedge (R \rightarrow S), P \vee R \therefore Q \vee S$	Constructive Dilemma
$P \rightarrow Q, Q \therefore P$ $P \rightarrow Q, Q \therefore P$	Modus Ponens	$(P \rightarrow Q) \wedge (R \rightarrow S), \neg Q \vee \neg S \therefore \neg P \vee \neg R$ $(P \rightarrow Q) \wedge (R \rightarrow S), \neg Q \vee \neg S \therefore \neg P \vee \neg R$	Destructive Dilemma
$P \rightarrow Q, \neg Q \therefore \neg P$ $P \rightarrow Q, \neg Q \therefore \neg P$	Modus Tollens		

Addition

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

Example

Let P be the proposition, "He studies very hard" is true

Therefore – "Either he studies very hard Or he is a very bad student." Here Q is the proposition "he is a very bad student".

Conjunction

If P and Q are two premises, we can use Conjunction rule to derive $P \wedge Q$.
 $P, Q \vdash P \wedge Q$

Example

Let P – “He studies very hard”

Let Q – “He is the best boy in the class”

Therefore – “He studies very hard and he is the best boy in the class”

Simplification

If $P \wedge Q$ is a premise, we can use Simplification rule to derive P.
 $P \wedge Q \vdash P$

Example

“He studies very hard and he is the best boy in the class”, $P \wedge Q$

Therefore – “He studies very hard”

Modus Ponens

If P and $P \rightarrow Q$ are two premises, we can use Modus Ponens to derive Q.
 $P, P \rightarrow Q \vdash Q$

Example

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

“You have a password”, P

Therefore – “You can log on to facebook”

Modus Tollens

If $P \rightarrow Q$ and $\neg Q$ are two premises, we can use Modus Tollens to derive $\neg P$.
 $P \rightarrow Q, \neg Q \vdash \neg P$

Example

“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 ”

Disjunctive Syllogism

If $\neg P \rightarrow \neg P$ and $P \vee Q, P \vee Q$ are two premises, we can use Disjunctive Syllogism to derive Q .
 $\neg P \vee Q :: Q \rightarrow \neg P \vee Q :: Q$

Example

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

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

Therefore - "The ice cream is chocolate flavored"

Hypothetical Syllogism

If $P \rightarrow Q, P \rightarrow Q$ and $Q \rightarrow R, Q \rightarrow R$ are two premises, we can use Hypothetical Syllogism to derive $P \rightarrow R, P \rightarrow R$

$$P \rightarrow Q, Q \rightarrow R :: P \rightarrow R, P \rightarrow Q, Q \rightarrow R :: P \rightarrow R$$

Example

"If it rains, I shall not go to school", $P \rightarrow Q, P \rightarrow Q$

"If I don't go to school, I won't need to do homework", $Q \rightarrow R, Q \rightarrow R$

Therefore - "If it rains, I won't need to do homework"

Constructive Dilemma

If $(P \rightarrow Q) \wedge (R \rightarrow S), (P \rightarrow Q) \wedge (R \rightarrow S)$ and $P \vee R, P \vee R$ are two premises, we can use constructive dilemma to derive $Q \vee S, Q \vee S$.

$$(P \rightarrow Q) \wedge (R \rightarrow S), P \vee R :: Q \vee S, (P \rightarrow Q) \wedge (R \rightarrow S), P \vee R :: Q \vee S$$

Example

"If it rains, I will take a leave", $(P \rightarrow Q), (P \rightarrow Q)$

"If it is hot outside, I will go for a shower", $(R \rightarrow S), (R \rightarrow S)$

"Either it will rain or it is hot outside", $P \vee R, P \vee R$

Therefore - "I will take a leave or I will go for a shower"

Destructive Dilemma

If $(P \rightarrow Q) \wedge (R \rightarrow S), (P \rightarrow Q) \wedge (R \rightarrow S)$ and $\neg Q \vee \neg S, \neg Q \vee \neg S$ are two premises, we can use destructive dilemma to derive $\neg P \vee \neg R, \neg P \vee \neg R$.

$$(P \rightarrow Q) \wedge (R \rightarrow S), \neg Q \vee \neg S :: \neg P \vee \neg R, (P \rightarrow Q) \wedge (R \rightarrow S), \neg Q \vee \neg S :: \neg P \vee \neg R$$

Example

"If it rains, I will take a leave", $(P \rightarrow Q), (P \rightarrow Q)$

"If it is hot outside, I will go for a shower", $(R \rightarrow S), (R \rightarrow S)$

"Either I will not take a leave or I will not go for a shower", $\neg Q \vee \neg S, \neg Q \vee \neg S$

Therefore - "Either it does not rain or it is not hot outside"

Predicates and quantifiers

Predicate Logic deals with predicates, which are propositions, consist of variables.

Predicate Logic - Definition

A predicate is an expression of one or more variables determined on some specific domain. A predicate with variables can be made a proposition by either authorizing a value to the variable or by quantifying the variable.

The following are some examples of predicates.

- Consider $E(x, y)$ denote "x = y"
- Consider $X(a, b, c)$ denote "a + b + c = 0"
- Consider $M(x, y)$ denote "x is married to y."

Quantifier:

The variable of predicates is quantified by quantifiers. There are two types of quantifier in predicate logic - Existential Quantifier and Universal Quantifier.

Existential Quantifier:

If $p(x)$ is a proposition over the universe U . Then it is denoted as $\exists x p(x)$ and read as "There exists at least one value in the universe of variable x such that $p(x)$ is true. The quantifier \exists is called the existential quantifier.

There are several ways to write a proposition, with an existential quantifier, i.e.,

$(\exists x \in A)p(x)$ or $\exists x \in A$ such that $p(x)$ or $(\exists x)p(x)$ or $p(x)$ is true for some $x \in A$.

Universal Quantifier:

If $p(x)$ is a proposition over the universe U . Then it is denoted as $\forall x, p(x)$ and read as "For every $x \in U, p(x)$ is true." The quantifier \forall is called the Universal Quantifier.

There are several ways to write a proposition, with a universal quantifier.

$\forall x \in A, p(x)$ or $p(x), \forall x \in A$ Or $\forall x, p(x)$ or $p(x)$ is true for all $x \in A$.

Negation of Quantified Propositions:

When we negate a quantified proposition, i.e., when a universally quantified proposition is negated, we obtain an existentially quantified proposition, and when an existentially quantified proposition is negated, we obtain a universally quantified proposition.

The two rules for negation of quantified proposition are as follows. These are also called DeMorgan's Law.

Example: Negate each of the following propositions:

$$1. \forall x \forall y p(x) \wedge \exists y q(y)$$

$$\begin{aligned} \text{Sol: } & \sim \forall x \forall y p(x) \wedge \exists y q(y) \\ & \cong \sim \forall x p(x) \vee \sim \exists y q(y) & (\because \sim(p \wedge q) = \sim p \vee \sim q) \\ & \cong \exists x \sim p(x) \vee \forall y \sim q(y) \end{aligned}$$

$$2. (\exists x \in U) (x+6=25)$$

$$\begin{aligned} \text{Sol: } & \sim (\exists x \in U) (x+6=25) \\ & \cong \forall x \in U \sim (x+6)=25 \\ & \cong (\forall x \in U) (x+6) \neq 25 \end{aligned}$$

$$3. \sim (\exists x p(x) \vee \forall y q(y))$$

$$\begin{aligned} \text{Sol: } & \sim (\exists x p(x) \vee \forall y q(y)) \\ & \cong \sim \exists x p(x) \wedge \sim \forall y q(y) & (\because \sim(p \vee q) = \sim p \wedge \sim q) \\ & \cong \forall x \sim p(x) \wedge \exists y \sim q(y) \end{aligned}$$

Propositions with Multiple Quantifiers:

The proposition having more than one variable can be quantified with multiple quantifiers. The multiple universal quantifiers can be arranged in any order without altering the meaning of the resulting proposition. Also, the multiple existential quantifiers can be arranged in any order without altering the meaning of the proposition.

The proposition which contains both universal and existential quantifiers, the order of those quantifiers can't be exchanged without altering the meaning of the proposition, e.g., the proposition $\exists x \forall y p(x,y)$ means "There exists some x such that $p(x,y)$ is true for every y ."

Example: Write the negation for each of the following. Determine whether the resulting statement is true or false. Assume $U = \mathbb{R}$.

$$1. \forall x \exists m (x^2 < m)$$

Sol: Negation of $\forall x \exists m (x^2 < m)$ is $\exists x \forall m (x^2 \geq m)$. The meaning of $\exists x \forall m (x^2 \geq m)$ is that there exists for some x such that $x^2 \geq m$, for every m . The statement is true as there is some greater x such that $x^2 \geq m$, for every m .

$$2. \exists m \forall x (x^2 < m)$$

Sol: Negation of $\exists m \forall x (x^2 < m)$ is $\forall m \exists x (x^2 \geq m)$. The meaning of $\forall m \exists x (x^2 \geq m)$ is that for every m , there exists for some x such that $x^2 \geq m$. The statement is true as for every m , there exists for some greater x such that $x^2 \geq m$.

Posets

There has been some research between posets and algebraic structures. Neggers [1] proved that there is a natural isomorphism between the category of pogroupoids and the category of posets. Neggers and Kim [2] showed that a poset (X, \leq) is $(C2+1-)$ -free if and only if its associated pogroupoid (X, \cdot) is modular*. Neggers and Kim [3] introduced the notion of d -algebras. It is a kind of generalization of BCK-algebras to which they discussed some relations between d -algebras and BCK-algebras, as well as some relations between d -algebras and oriented digraphs. Cha et al. [4] introduced the notions of a trend and probability functions on d -algebras. They obtained an equivalent condition defining a trend π_0 with condition (j) on a standard BCK-algebra. Loof et al. [5] discussed mutual rank probabilities in partially ordered sets. Baets et al. [6] characterized the transitivity of the mutual rank probability relation of a poset, and Lerche et al. [7] evaluated ranking probabilities for partial orders based on random linear extensions.

In this paper, we define a probability function on a poset. The idea of a probability function on a poset came from [4], and we obtained some probability functions on a poset. We defined a probability realizer on a poset, and found some examples for probability realizers of posets for the standard probability function π_0 . Moreover, we applied the notion of a probability function to the ordered plane (order geometry), and found three probability functions acting on it. Some comments have been suggested for further research.

Preliminaries

Some definitions and terminologies will be recalled for partially ordered sets which are necessary for reading this paper.

An ordered pair (X, \leq) is called a partially ordered set if \leq is a partial order, i.e., reflexive, anti-symmetric, and transitive, on the set X . A poset (X, \leq) is said to be a chain if every two distinct elements of X are comparable, and we denote it by C_n when the cardinality of X is equal to n . A poset (X, \leq) is said to be an anti-chain if every two distinct elements of X are incomparable, and we denote it by $n--$ when the cardinality of X is equal to n . Given two posets X and Y , a poset Z is said to be an ordinal sum of X and Y if $z_1 \leq z_2$ in Z , then either $z_1 \in X$ and $z_2 \in Y$ or $z_1 \leq z_2$ in X ,

or $z_1 \leq z_2$ in Y . A graph Z can be realized by placing the Hasse diagram of Y above the Hasse diagram of X , and by drawing line segments from the maximal elements of X to all the minimal elements of Y . We denote it by $Z = X \oplus Y$. A chain (X, \leq^*) is said to be a linear extension of a poset (X, \leq) if $x \leq y$ implies $x \leq^* y$. A family of linear extensions $R = \{L_1, \dots, L_k\}$ of a poset (X, \leq) is said to be a realizer of (X, \leq) if (X, \leq) can be realized as the intersection of R , but not as the intersection of fewer than k linear extensions. For details we refer to [8].

A non-empty set X with a constant 0 and a binary operation “ $*$ ” is said to be a d-algebra [3] if it satisfies: (i) $x * x = 0$, (ii) $0 * x = 0$, and (iii) $x * y = 0$ and $y * x = 0$ imply $x = y$ for all $x, y \in X$.

A mapping $\pi: X \times X \rightarrow [0, 1]$ is said to be a trend [4] on a d-algebra $(X, *, 0)$ if it satisfies: for any $x, y \in X$,

$$x * y = 0 \text{ implies } \pi(x, y) = 1, \tag{a}$$

$$x * y \neq 0 \text{ implies } \pi(x, y) + \pi(y, x) = 1. \tag{b}$$

A trend $\pi: X \times X \rightarrow [0, 1]$ is said to be a probability function [4] on a d-algebra $(X, *, 0)$ if it satisfies: for any $x, y, z \in X$,

$$y * z = 0 \text{ implies } \pi(x, y) \leq \pi(x, z). \tag{c}$$

It is of course possible to consider other conditions to build (different) notions of trends and probability functions and to compare the resulting classes with those obtained here. In fact, we will actually do so below. As an example of the situation above, let $X := [0, \infty)$ and let $x * y := 0$ if and only if $x \leq y$, and $x * y := 1$ otherwise. Thus (a) holds if $\pi(x, y) := x * y$. If $y \leq z$, then $x \leq y$ implies $x \leq z$ while $y > z$ means $\pi(y, z) = y * z = 1$, so that (c) does not apply in that case. Condition (b) holds since $x * y \neq 0$ implies $x * y = 1$, and $x > y$ yields $\pi(y, x) = 1$ since $y < x$ in that case. The groupoid $(X, *, 0)$ is certainly a d-algebra.

Hasse Diagram

It is a useful tool, which completely describes the associated partial order. Therefore, it is also called an ordering diagram. It is very easy to convert a directed graph of a relation on a set A to an equivalent Hasse diagram. Therefore, while drawing a Hasse diagram following points must be remembered.

1. The vertices in the Hasse diagram are denoted by points rather than by circles.
2. Since a partial order is reflexive, hence each vertex of A must be related to itself, so the edges from a vertex to itself are deleted in Hasse diagram.
3. Since a partial order is transitive, hence whenever aRb , bRc , we have aRc . Eliminate all edges that are implied by the transitive property in Hasse diagram, i.e., Delete edge from a to c but retain the other two edges.
4. If a vertex ' a ' is connected to vertex ' b ' by an edge, i.e., aRb , then the vertex ' b ' appears above vertex ' a '. Therefore, the arrow may be omitted from the edges in the Hasse diagram.

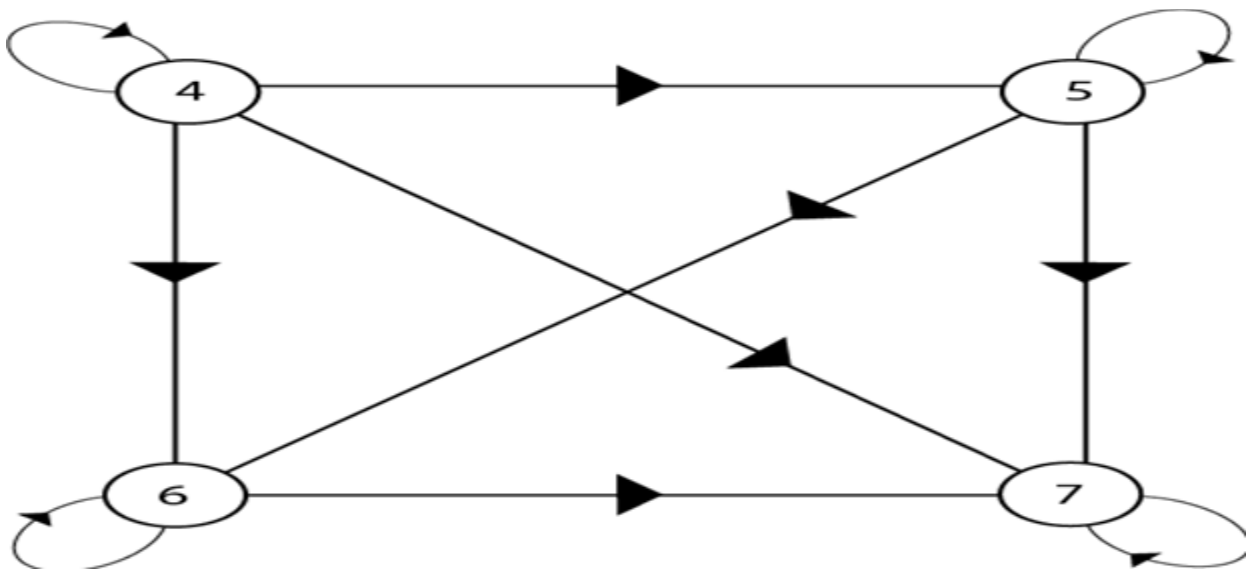
The Hasse diagram is much simpler than the directed graph of the partial order.

Example: Consider the set $A = \{4, 5, 6, 7\}$. Let R be the relation \leq on A . Draw the directed graph and the Hasse diagram of R .

Solution: The relation \leq on the set A is given by

$$R = \{\{4, 5\}, \{4, 6\}, \{4, 7\}, \{5, 6\}, \{5, 7\}, \{6, 7\}, \{4, 4\}, \{5, 5\}, \{6, 6\}, \{7, 7\}\}$$

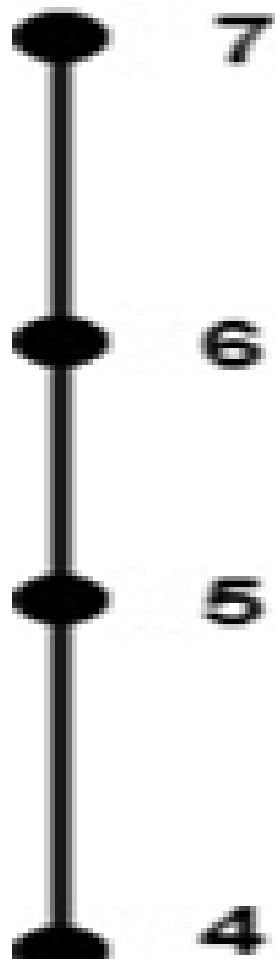
The directed graph of the relation R is as shown in fig:



To draw the Hasse diagram of partial order, apply the following points:

1. Delete all edges implied by reflexive property i.e.
(4, 4), (5, 5), (6, 6), (7, 7)
2. Delete all edges implied by transitive property i.e.
(4, 7), (5, 7), (4, 6)
3. Replace the circles representing the vertices by dots.
4. Omit the arrows.

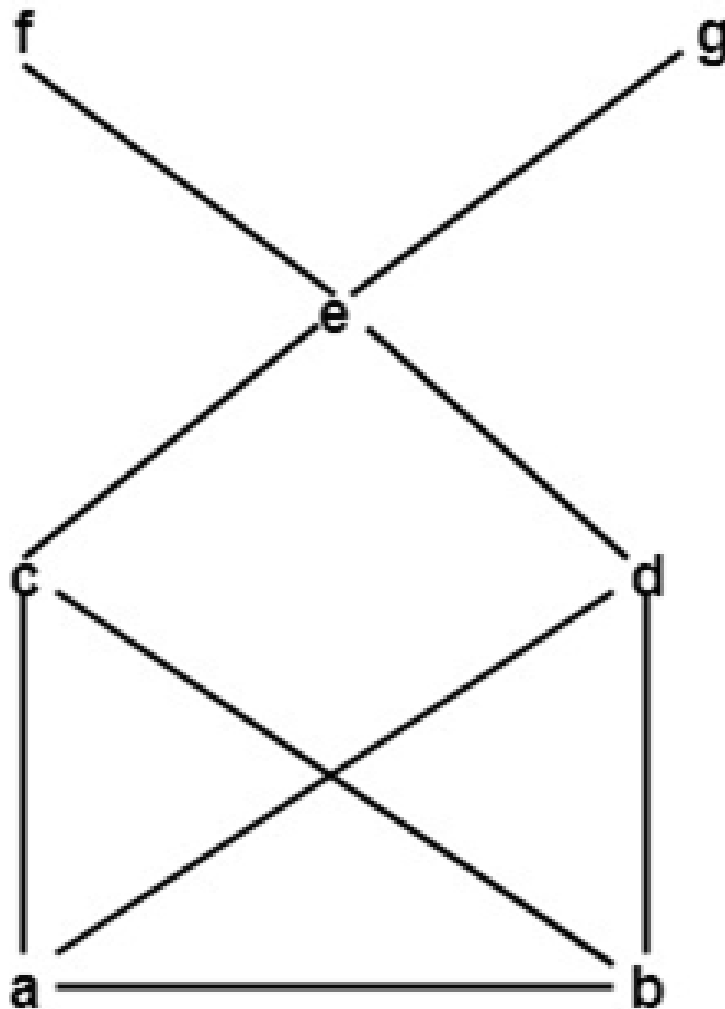
The Hasse diagram is as shown in fig:



Upper Bound: Consider B be a subset of a partially ordered set A. An element $x \in A$ is called an upper bound of B if $y \leq x$ for every $y \in B$.

Lower Bound: Consider B be a subset of a partially ordered set A. An element $z \in A$ is called a lower bound of B if $z \leq x$ for every $x \in B$.

Example: Consider the poset $A = \{a, b, c, d, e, f, g\}$ be ordered shown in fig. Also let $B = \{c, d, e\}$. Determine the upper and lower bound of B.



Solution: The upper bound of B is e, f, and g because every element of B is ' \leq ' e, f, and g.

The lower bounds of B are a and b because a and b are ' \leq ' every elements of B.

Least Upper Bound (SUPREMUM):

Let A be a subset of a partially ordered set S. An element M in S is called an upper bound of A if M succeeds every element of A, i.e. if, for every x in A, we have $x \leq M$

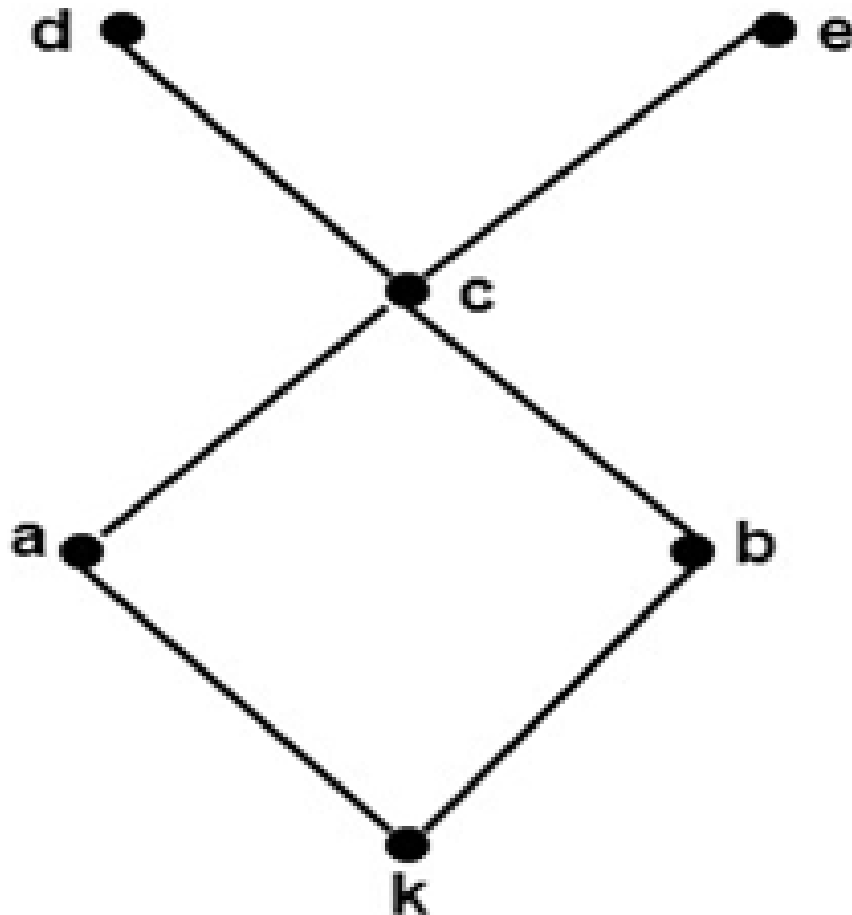
If an upper bound of A precedes every other upper bound of A, then it is called the supremum of A and is denoted by $\text{Sup}(A)$

Greatest Lower Bound (INFIMUM):

An element m in a poset S is called a lower bound of a subset A of S if m precedes every element of A , i.e. if, for every y in A , we have $m \leq y$

If a lower bound of A succeeds every other lower bound of A , then it is called the infimum of A and is denoted by $\text{Inf}(A)$

Example: Determine the least upper bound and greatest lower bound of $B = \{a, b, c\}$ if they exist, of the poset whose Hasse diagram is shown in fig:



Solution: The least upper bound is c .

The greatest lower bound is k .

Lattices:

Introduction

Let L be a non-empty set closed under two binary operations called meet and join, denoted by \wedge and \vee . Then L is called a lattice if the following axioms hold where a, b, c are elements in L :

1) Commutative Law: -

$$(a) a \wedge b = b \wedge a \quad (b) a \vee b = b \vee a$$

2) Associative Law:-

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

3) Absorption Law: -

$$(a) a \wedge (a \vee b) = a \quad (b) a \vee (a \wedge b) = a$$

Duality:

The dual of any statement in a lattice (L, \wedge, \vee) is defined to be a statement that is obtained by interchanging \wedge and \vee .

For example, the dual of $a \wedge (b \vee a) = a \vee a$ is $a \vee (b \wedge a) = a \wedge a$

Bounded Lattices:

A lattice L is called a bounded lattice if it has greatest element 1 and a least element 0.

Example:

1. The power set $P(S)$ of the set S under the operations of intersection and union is a bounded lattice since \emptyset is the least element of $P(S)$ and the set S is the greatest element of $P(S)$.
2. The set of +ve integer I_+ under the usual order of \leq is not a bounded lattice since it has a least element 1 but the greatest element does not exist.

Properties of Bounded Lattices:

If L is a bounded lattice, then for any element $a \in L$, we have the following identities:

1. $a \vee 1 = 1$
2. $a \wedge 1 = a$
3. $a \vee 0 = a$
4. $a \wedge 0 = 0$

Theorem: Prove that every finite lattice $L = \{a_1, a_2, a_3, \dots, a_n\}$ is bounded.

Proof: We have given the finite lattice:

$$L = \{a_1, a_2, a_3, \dots, a_n\}$$

Thus, the greatest element of Lattices L is $a_1 \vee a_2 \vee a_3 \vee \dots \vee a_n$.

Also, the least element of lattice L is $a_1 \wedge a_2 \wedge a_3 \wedge \dots \wedge a_n$.

Since, the greatest and least elements exist for every finite lattice. Hence, L is bounded.

Sub-Lattices:

Consider a non-empty subset L_1 of a lattice L. Then L_1 is called a sub-lattice of L if L_1 itself is a lattice i.e., the operation of L i.e., $a \vee b \in L_1$ and $a \wedge b \in L_1$ whenever $a \in L_1$ and $b \in L_1$.

Example: Consider the lattice of all +ve integers I_+ under the operation of divisibility. The lattice D_n of all divisors of $n > 1$ is a sub-lattice of I_+ .

Determine all the sub-lattices of D_{30} that contain at least four elements, $D_{30} = \{1, 2, 3, 5, 6, 10, 15, 30\}$.

Solution: The sub-lattices of D_{30} that contain at least four elements are as follows:

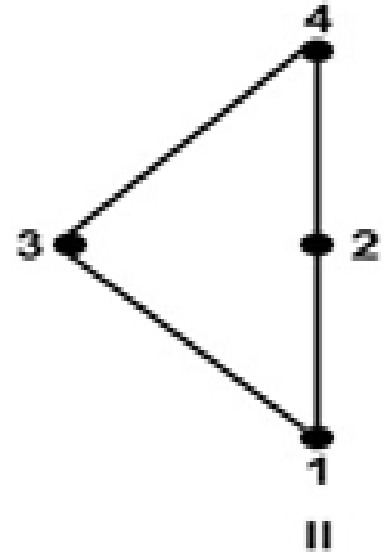
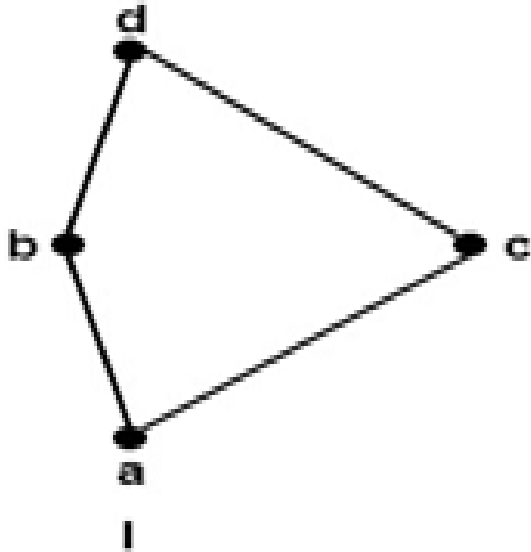
1. $\{1, 2, 6, 30\}$
2. $\{1, 2, 3, 30\}$
3. $\{1, 5, 15, 30\}$
4. $\{1, 3, 6, 30\}$
5. $\{1, 5, 10, 30\}$
6. $\{1, 3, 15, 30\}$
7. $\{2, 6, 10, 30\}$

Isomorphic Lattices:

Two lattices L_1 and L_2 are called isomorphic lattices if there is a bijection from L_1 to L_2 i.e., $f: L_1 \rightarrow L_2$, such that $f(a \wedge b) = f(a) \wedge f(b)$ and $f(a \vee b) = f(a) \vee f(b)$

Example: Determine whether the lattices shown in fig are isomorphic.

Solution: The lattices shown in fig are isomorphic. Consider the mapping $f = \{(a, 1), (b, 2), (c, 3), (d, 4)\}$. For example $f(b \wedge c) = f(a) = 1$. Also, we have $f(b) \wedge f(c) = 2 \wedge 3 = 1$



Distributive Lattice:

A lattice L is called distributive lattice if for any elements a, b and c of L, it satisfies following distributive properties:

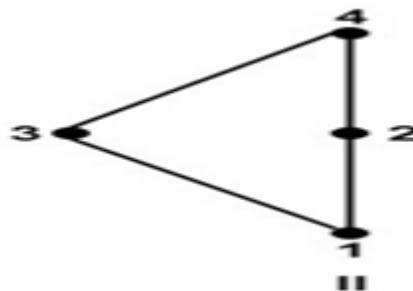
1. $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
2. $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

If the lattice L does not satisfies the above properties, it is called a non-distributive lattice.

Example:

1. The power set P (S) of the set S under the operation of intersection and union is a distributive function. Since,

$$a \cap (b \cup c) = (a \cap b) \cup (a \cap c)$$
 and, also $a \cup (b \cap c) = (a \cup b) \cap (a \cup c)$ for any sets a, b and c of P(S).
2. The lattice shown in fig II is a distributive. Since, it satisfies the distributive properties for all ordered triples which are taken from 1, 2, 3, and 4.

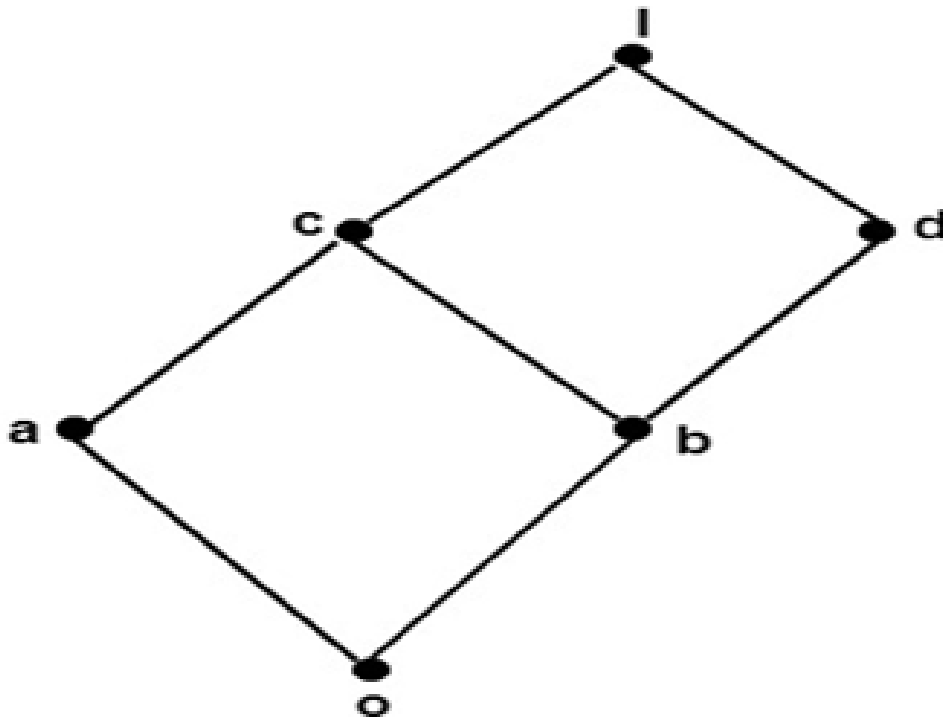


Complements and complemented lattices:

Let L be a bounded lattice with lower bound o and upper bound l . Let a be an element of L . An element x in L is called a complement of a if $a \vee x = l$ and $a \wedge x = o$

A lattice L is said to be complemented if L is bounded and every element in L has a complement.

Example: Determine the complement of a and c in fig:



Solution: The complement of a is d . Since, $a \vee d = 1$ and $a \wedge d = 0$

The complement of c does not exist. Since, there does not exist any element c' such that $c \vee c' = 1$ and $c \wedge c' = 0$.

Modular Lattice:

A lattice (L, \wedge, \vee) is called a modular lattice if $a \vee (b \wedge c) = (a \vee b) \wedge c$ whenever $a \leq c$.

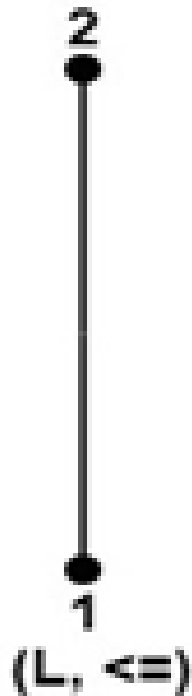
Direct Product of Lattices:

Let (L_1, \vee_1, \wedge_1) and (L_2, \vee_2, \wedge_2) be two lattices. Then (L, \wedge, \vee) is the direct product of lattices, where $L = L_1 \times L_2$ in which the binary operation \vee (join) and \wedge (meet) on L are such that for any (a_1, b_1) and (a_2, b_2) in L .

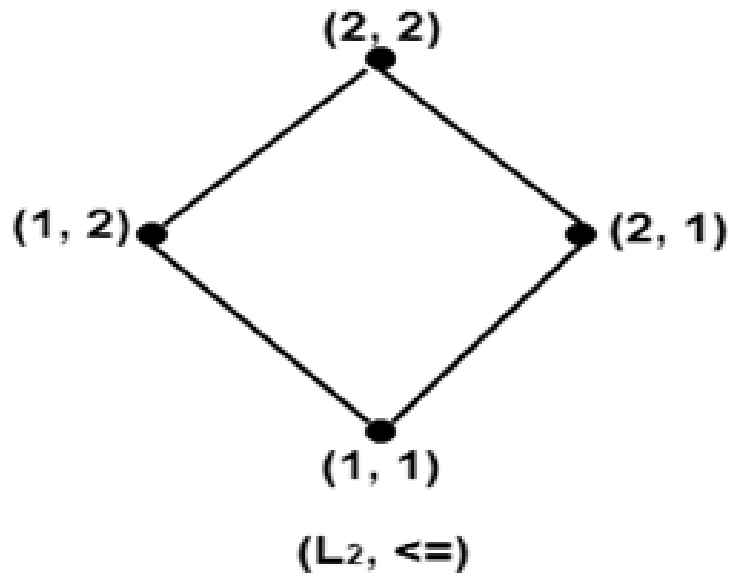
$$(a_1, b_1) \vee (a_2, b_2) = (a_1 \vee_1 a_2, b_1 \vee_2 b_2)$$

and $(a_1, b_1) \wedge (a_2, b_2) = (a_1 \wedge_1 a_2, b_1 \wedge_2 b_2).$

Example: Consider a lattice (L, \leq) as shown in fig. where $L = \{1, 2\}$. Determine the lattices (L^2, \leq) , where $L^2 = L \times L$.



Solution: The lattice (L^2, \leq) is shown in fig:



Ordered set

A lattice-ordered set is a poset (L, \leq) in which each two-element subset $\{a, b\}$ has an infimum, denoted $\inf \{a, b\}$, and a supremum, denoted $\sup \{a, b\}$. There is a natural relationship between lattice-ordered sets and lattices. In fact, a lattice (L, \wedge, \vee) is obtained from a lattice-ordered poset (L, \leq) by defining $a \wedge b = \inf \{a, b\}$ and $a \vee b = \sup \{a, b\}$ for any $a, b \in L$. Also, from a lattice (L, \wedge, \vee) , one may obtain a lattice-ordered set (L, \leq) by setting $a \leq b$ in L if and only if $a = a \wedge b$. One obtains the same lattice-ordered set (L, \leq) from the given lattice by setting $a \leq b$ in L if and only if $a \vee b = b$. (In other words, one may prove that for any lattice, (L, \wedge, \vee) , and for any two members a and b of L , $a \wedge b = b$ if and only if $a = a \vee b$.)

Lattice-ordered sets abound in mathematics and its applications, and many authors do not distinguish between them and lattices. From a universal algebraist's point of view, however, a lattice is different from a lattice-ordered set because lattices are algebraic structures that form an equational class or variety, but lattice-ordered sets are not algebraic structures, and therefore do not form a variety.

A lattice-ordered set is bounded provided that it is a bounded poset, i.e., if it has an upper bound and a lower bound. For a bounded lattice-ordered set, the upper bound is frequently denoted 1 and the lower bound is frequently denoted 0. Given an element x of a bounded lattice-ordered set (L, \leq) , we say that x is complemented in (L, \leq) if there exists an element

$y \in L$ such that $\inf \{x, y\} = 0$ and $\sup \{x, y\} = 1$

Hasse diagram of partially ordered set

Consider a relation R on a set S satisfying the following properties:

1. R is reflexive, i.e., xRx for every $x \in S$.
2. R is antisymmetric, i.e., if xRy and yRx , then $x = y$.
3. R is transitive, i.e., xRy and yRz , then xRz .

Then R is called a partial order relation, and the set S together with partial order is called a partially order set or POSET and is denoted by (S, \leq) .

Example:

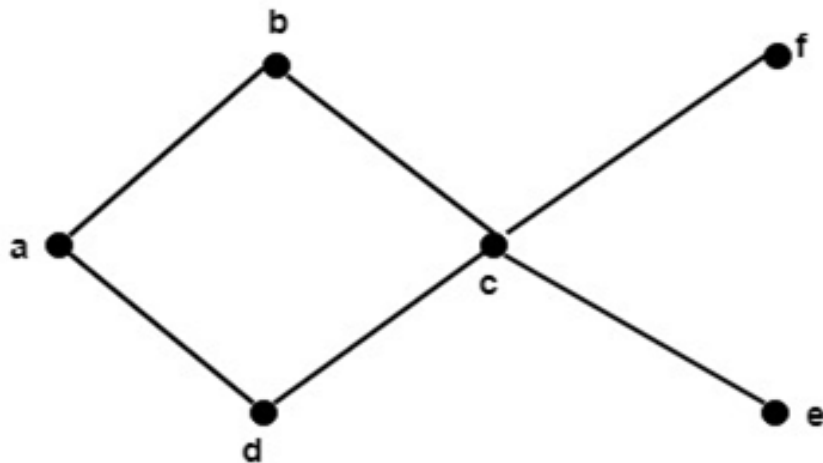
1. The set N of natural numbers form a poset under the relation ' \leq ' because firstly $x \leq x$, secondly, if $x \leq y$ and $y \leq x$, then we have $x = y$ and lastly if $x \leq y$ and $y \leq z$, it implies $x \leq z$ for all $x, y, z \in N$.

2. The set N of natural numbers under divisibility i.e., 'x divides y' forms a poset because x/x for every $x \in N$. Also if x/y and y/x , we have $x = y$. Again if x/y , y/z we have x/z , for every $x, y, z \in N$.
3. Consider a set $S = \{1, 2\}$ and power set of S is $P(S)$. The relation of set inclusion \subseteq is a partial order. Since, for any sets A, B, C in $P(S)$, firstly we have $A \subseteq A$, secondly, if $A \subseteq B$ and $B \subseteq A$, then we have $A = B$. Lastly, if $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$. Hence, $(P(S), \subseteq)$ is a poset.

Elements of POSET:

1. **Maximal Element:** An element $a \in A$ is called a maximal element of A if there is no element in c in A such that $a \leq c$.
2. **Minimal Element:** An element $b \in A$ is called a minimal element of A if there is no element in c in A such that $c \leq b$.

Example: Determine all the maximal and minimal elements of the poset whose Hasse diagram is shown in fig:



Solution: The maximal elements are b and f .

The minimal elements are d and e .

Comparable Elements:

Consider an ordered set A . Two elements a and b of set A are called comparable if

$$a \leq_R b \quad \text{or} \quad b \leq_R a$$

Non-Comparable Elements:

Consider an ordered set A . Two elements a and b of set A are called non-comparable if neither $a \leq b$ nor $b \leq a$.

Example: Consider $A = \{1, 2, 3, 5, 6, 10, 15, 30\}$ is ordered by divisibility. Determine all the comparable and non-comparable pairs of elements of A .

Solution: The comparable pairs of elements of A are:

$\{1, 2\}, \{1, 3\}, \{1, 5\}, \{1, 6\}, \{1, 10\}, \{1, 15\}, \{1, 30\}$
 $\{2, 6\}, \{2, 10\}, \{2, 30\}$
 $\{3, 6\}, \{3, 15\}, \{3, 30\}$
 $\{5, 10\}, \{5, 15\}, \{5, 30\}$
 $\{6, 30\}, \{10, 30\}, \{15, 30\}$

The non-comparable pair of elements of A are:

$\{2, 3\}, \{2, 5\}, \{2, 15\}$
 $\{3, 5\}, \{3, 10\}, \{5, 6\}, \{6, 10\}, \{6, 15\}, \{10, 15\}$

Linearly Ordered Set:

Consider an ordered set A . The set A is called linearly ordered set or totally ordered set, if every pair of elements in A is comparable.

Example: The set of positive integers I_+ with the usual order \leq is a linearly ordered set.

Consistent enumeration

An **enumeration** is a complete, ordered listing of all the items in a collection. The term is commonly used in mathematics and computer science to refer to a listing of all of the elements of a set. The precise requirements for an enumeration (for example, whether the set must be finite, or whether the list is allowed to contain repetitions) depend on the discipline of study and the context of a given problem.

Some sets can be enumerated by means of a **natural ordering** (such as 1, 2, 3, 4, ... for the set of positive integers), but in other cases it may be necessary to impose a (perhaps arbitrary) ordering. In some contexts, such as enumerative combinatorics, the term enumeration is used more in the sense of counting – with emphasis on determination of the number of elements that a set contains, rather than the production of an explicit listing of those elements.

In combinatorics, enumeration means counting, i.e., determining the exact number of elements of finite sets, usually grouped into infinite families, such as the family of sets

each consisting of all permutations of some finite set. There are flourishing subareas in many branches of mathematics concerned with enumerating in this sense objects of special kinds. For instance, in partition enumeration and graph enumeration the objective is to count partitions or graphs that meet certain conditions.

Isomorphic ordered set

Let P be a set and \sqsubseteq be a (partial) order on P . Then P and \sqsubseteq form a (*partially*) *ordered set*.

If the order is total, so that no two elements of P are incomparable, then the ordered set is a *totally ordered set*. Totally ordered sets are the ones people are first familiar with. See Figure 1 for an example.

A totally ordered set is also termed a *chain*.

If the order is partial, so that P has two or more incomparable elements, then the ordered set is a *partially ordered set*. See Figure 2 for an example.

At the other extreme, if no two elements are comparable unless they are equal, then the ordered set is an *antichain*. See Figure 3.

On any set, $=$ is an order; this is termed the *discrete order* on the set. Any set ordered by $=$ forms an antichain.

It is common for people to refer briefly though inaccurately to an ordered set as an *order*, to a totally ordered set as a *total order*, and to a partially ordered set as a *partial order*. It is usually clear by context whether "order" refers literally to an order (an order relation) or by synecdoche to an ordered set.

Examples:

1. The integers with \leq form an ordered set (see Figure 1). \leq is a total order on the integers, so this ordered set is a chain.
2. Any powerset with \subseteq forms an ordered set (see Figure 2). This is a partially ordered set because not all subsets are related by \subseteq , for example $\{a\} \not\subseteq \{b, r\}$.
3. A set of unrelated items, ordered by $=$, is the discrete order on that set and forms an antichain (see Figure 3).
4. The classes in `java.util` with the subclass relation form an ordered set (see Figure 4). This set is partially ordered, because not all classes in the set are related by the subclass relations (for example, `Vector` and `HashSet` are not related and are thus incomparable: `Vector` $\not\subseteq$ `HashSet`).

5. A set of binary strings with the prefix relation forms an ordered set (see Figure 5). This set is partially ordered because not all strings are related by the prefix relation, for example $01 \parallel 10$.
6. The (non-empty) conjunctions of any of the propositions p , q , and r , ordered by implication, form an ordered set (see Figure 6). In this set, $p \wedge q$ implies q , but $p \wedge q$ neither implies nor is implied by $q \wedge r$, so $p \wedge q$ and $q \wedge r$ are incomparable ($p \wedge q \# q \wedge r$).
7. The positive integers N with the divisibility relation form an ordered set. The divisibility relation relates m to n if m divides n , written $m \mid n$. Thus $2 \mid 6$, and $3 \mid 6$ but not $4 \mid 6$ (i.e., 4 and 6 are incomparable, written $4 \parallel 6$) because 4 does not divide 6. And for any $n \in N$, $1 \mid n$ and $n \mid n$. A part of this ordered set is shown in Figure 7.

Well ordered set

A set P equipped with a binary relation \leq that satisfies the following conditions:

1. For any $x, y \in P$, either $x \leq y$ or $y \leq x$.
2. For any $x, y \in P$, if $x \leq y$ and $y \leq x$, then $x = y$.
3. For any $x, y, z \in P$, if $x \leq y$ and $y \leq z$, then $x \leq z$.
4. In any non-empty subset $X \subseteq P$, there exists an element a such that $a \leq x$ for all $x \in X$.

Thus, a well-ordered set is a totally ordered set satisfying the minimum condition.

The concept of a well-ordered set was introduced by G. Cantor ([1]). An example of a well-ordered set is the naturally ordered set of natural numbers. On the other hand, the interval of real numbers $[0, 1]$ with the natural order is not well-ordered. Any subset of a well-ordered set is itself well-ordered. The Cartesian product of a finite number of well-ordered sets is well-ordered by the relation of lexicographic order. A totally ordered set is well-ordered if and only if it contains no subset that is anti-isomorphic to the set of natural numbers.

The smallest element of a well-ordered set P is denoted by zero (the symbol 0). For any element $a \in P$, the set

$$[0, a) = \{x \in P, x < a\}$$

is called an initial segment of P . For any element a that is not the largest element in P , there exists an element immediately following it; it is usually denoted by $a+1$.

An element of a well-ordered set that has no element immediately preceding it is called a limit element.

The Comparison Theorem.

For any two well-ordered sets P_1 and P_2 , one and only one of the following situations occurs: (a) P_1 is isomorphic to P_2 ; (b) P_1 is isomorphic to an initial segment of P_2 ; or (c) P_2 is isomorphic to an initial segment of P_1 .

If the axiom of choice is included in the axioms of set theory, it may be shown that it is possible to impose on any non-empty set an order relation that converts it into a well-ordered set (i.e., any non-empty set can be well-ordered). This theorem, known as Zermelo's Well-Ordering Theorem, is in fact equivalent to the axiom of choice. Zermelo's Well-Ordering Theorem and the Comparison Theorem form the basis for the comparison between cardinalities of sets. Order types of well-ordered sets are called ordinal numbers.

References

- [1] G. Cantor, "Über unendliche, lineare Punktmannigfaltigkeiten", Math. Ann., **21** (1883), pp. 51–58.
- [2] P.S. Aleksandrov, "Einführung in die Mengenlehre und die Theorie der reellen Funktionen", Deutsch. Verlag Wissenschaft. (1956). (Translated from Russian)
- [3] F. Hausdorff, "Grundzüge der Mengenlehre", Leipzig (1914). (Reprinted (incomplete) English translation: Set theory, Chelsea (1978))
- [4] N. Bourbaki, "Elements of mathematics. Theory of sets", Addison-Wesley (1968). (Translated from French)
- [5] K. Kuratowski, A. Mostowski, "Set theory", North-Holland (1968).

Comments

In the definition above, Condition (3) (the transitivity of the order relation) is in fact redundant: It follows from the existence of a least element in the subset $\{x,y,z\}$.

Sometimes, a well-ordered set is called a **totally well-ordered set**, reflecting the fact that the ordering is a total ordering or linear ordering.

References

[a1] A. Levy, "Basic set theory", Springer (1979).

Properties of lattices

Man and nature both exploit the remarkable properties of cellular solids, by which we mean foams, meshes and microlattices. To the non-scientist, their image is that of soft, compliant, things: cushions, packaging and padding. To the food scientist they are familiar as bread, cake and desserts of the best kind: meringue, mousse and sponge. To those who study nature they are the structural materials of their subject: wood, coral, cancellous bone. And to the engineer they are of vast importance in building lightweight structures, for energy management, for thermal insulation, filtration and much more.

When a solid is converted into a material with a foam-like structure, the single-valued properties of the solid are extended. By properties we mean stiffness, strength, thermal conductivity and diffusivity, electrical resistivity and so forth. And the extension is vast—the properties can be changed by a factor of 1000 or more. Perhaps the most important concept in analysing the mechanical behaviour is that of the distinction between a *stretch*- and a *bending*-dominated structure. The first is exceptionally stiff and strong for a given mass; the second is compliant and, although not strong, it absorbs energy well when compressed. This paper summarizes a little of the way in which the mechanical properties of cellular solids are analysed and illustrates the range of properties offered by alternative configurations.

Bounded lattices

"Bounded" and "boundary" are distinct concepts; for the latter see boundary (topology). A circle in isolation is a boundaryless bounded set, while the half plane is unbounded yet has a boundary.

In mathematical analysis and related areas of mathematics, a set is called bounded if it is, in a certain sense, of finite size. Conversely, a set which is not bounded is called unbounded. The word 'bounded' makes no sense in a general topological space without a corresponding metric.

A set S of real numbers is called bounded from above if there exists some real number k (not necessarily in S) such that $k \geq s$ for all s in S . The number k is called an upper bound of S . The terms bounded from below and lower bound are similarly defined.

A set S is bounded if it has both upper and lower bounds. Therefore, a set of real numbers is bounded if it is contained in a finite interval.

Distributive lattices

In mathematics, a distributive lattice is a lattice in which the operations of join and meet distribute over each other. The prototypical examples of such structures are collections of sets for which the lattice operations can be given by set union and intersection. Indeed, these lattices of sets describe the scenery completely: every distributive lattice is—up to isomorphism—given as such a lattice of sets.

As in the case of arbitrary lattices, one can choose to consider a distributive lattice L either as a structure of order theory or of universal algebra. Both views and their mutual correspondence are discussed in the article on lattices. In the present situation, the algebraic description appears to be more convenient:

A lattice (L, \vee, \wedge) is distributive if the following additional identity holds for all x, y , and z in L :

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z).$$

Viewing lattices as partially ordered sets, this says that the meet operation preserves non-empty finite joins. It is a basic fact of lattice theory that the above condition is equivalent to its dual:^[1]

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z) \quad \text{for all } x, y, \text{ and } z \text{ in } L.^{[2]}$$

In every lattice, defining $p \leq q$ as usual to mean $p \wedge q = p$, the inequality $x \wedge (y \vee z) \geq (x \wedge y) \vee (x \wedge z)$ holds as well as its dual inequality $x \vee (y \wedge z) \leq (x \vee y) \wedge (x \vee z)$. A lattice is distributive if one of the converse inequalities holds, too. More information on the relationship of this condition to other distributivity conditions of order theory can be found in the article on distributivity (order theory).

A morphism of distributive lattices is just a lattice homomorphism as given in the article on lattices, i.e. a function that is compatible with the two lattice operations. Because such a morphism of lattices preserves the lattice structure, it will consequently also preserve the distributivity (and thus be a morphism of distributive lattices).

Complemented lattices

Let L be a bounded lattice (with 0 and 1), and $a \in L$. A complement of a is an element $b \in L$ such that

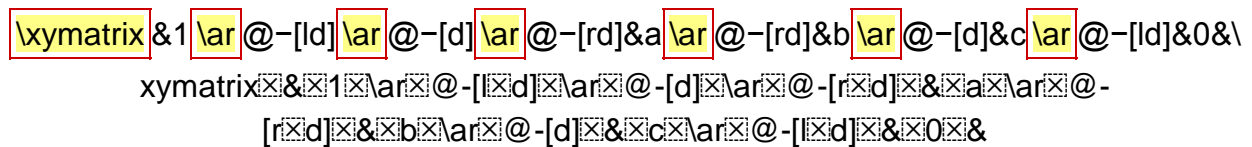
$$a \wedge b = 0 \quad \text{and} \quad a \vee b = 1.$$

Remark. Complements may not exist. If L is a non-trivial chain, then no element (other than 0 and 1) has a complement. This also shows that if a is a complement of a non-trivial element b , then a and b form an antichain.

An element in a bounded lattice is complemented if it has a complement. A complemented lattice is a bounded lattice in which every element is complemented.

Remarks.

In a complemented lattice, there may be more than one complement corresponding to each element. Two elements are said to be related, or perspective if they have a common complement. For example, the following lattice is complemented.



Note that none of the non-trivial elements have unique complements. Any two non-trivial elements are related via the third.

If a complemented lattice L is a distributive lattice, then L is uniquely complemented (in fact, a Boolean lattice). For if y_1 and y_2 are two complements of x , then

$$y_2 = 1 \wedge y_2 = (x \vee y_1) \wedge y_2 = (x \wedge y_2) \vee (y_1 \wedge y_2) = 0 \vee (y_1 \wedge y_2) = y_1 \wedge y_2. \quad y_2 = 1 \wedge y_2 = (x \vee y_1) \wedge y_2 = (x \wedge y_2) \vee (y_1 \wedge y_2) = 0 \vee (y_1 \wedge y_2) = y_1 \wedge y_2.$$

Similarly, $y_1 = y_2 \wedge y_1 = y_2 \wedge y_1$. So $y_2 = y_1 \wedge y_2 = y_1$.

In the category of complemented lattices, a morphism between two objects is a $\{0,1\}$ -lattice homomorphism; that is, a lattice homomorphism which preserves 00 and 11.

Title	complemented lattice
Canonical name	ComplementedLattice
Date of creation	2013-03-22 15:02:25
Last modified on	2013-03-22 15:02:25
Owner	CWoo (3771)
Last modified by	CWoo (3771)
Numerical id	26

Author	CWoo (3771)
Entry type	Definition
Classification	msc 06C15
Classification	msc 06B05
Synonym	perspective elements
Synonym	complemented
Related topic	Perspectivity
Related topic	OrthocomplementedLattice
Related topic	PseudocomplementedLattice
Related topic	DifferenceOfLatticeElements
Related topic	Pseudocomplement
Defines	related elements in lattice
Defines	complement
Defines	complemented element

UNIT-III

Introduction to defining language

In mathematics, computer science, and linguistics, a formal language consists of words whose letters are taken from an alphabet and are well-formed according to a specific set of rules.

The alphabet of a formal language consists of symbols, letters, or tokens that concatenate into strings of the language.^[1] Each string concatenated from symbols of this alphabet is called a word, and the words that belong to a particular formal language are sometimes called well-formed words or well-formed formulas. A formal language is often defined by means of a formal grammar such as a regular grammar or context-free grammar, which consists of its formation rules.

The field of formal language theory studies primarily the purely syntactical aspects of such languages—that is, their internal structural patterns. Formal language theory sprang out of linguistics, as a way of understanding the syntactic regularities of natural languages. In computer science, formal languages are used among others as the basis for defining the grammar of programming languages and formalized versions of subsets of natural languages in which the words of the language represent concepts that are associated with particular meanings or semantics. In computational complexity theory, decision problems are typically defined as formal languages, and complexity classes are defined as the sets of the formal languages that can be parsed by machines with limited computational power. In logic and the foundations of mathematics, formal languages are used to represent the syntax of axiomatic systems, and mathematical formalism is the philosophy that all of mathematics can be reduced to the syntactic manipulation of formal languages in this way.

The first formal language is thought to be the one used by Gottlob Frege in his *Begriffsschrift* (1879), literally meaning "concept writing", and which Frege described as a "formal language of pure thought".

Words over an alphabet

An alphabet, in the context of formal languages, can be any set, although it often makes sense to use an alphabet in the usual sense of the word, or more generally a character set such as ASCII or Unicode. The elements of an alphabet are called its letters. An alphabet may contain an infinite number of elements;^[note 1] however, most definitions in formal language theory specify alphabets with a finite number of elements, and most results apply only to them.

A word over an alphabet can be any finite sequence (i.e., string) of letters. The set of all words over an alphabet Σ is usually denoted by Σ^* (using the Kleene star). The length of a word is the number of letters it is composed of. For any alphabet, there is only one word of length 0, the empty word, which is often denoted by ϵ , λ or even Λ . By concatenation one can combine two words to form a new word, whose length is the

sum of the lengths of the original words. The result of concatenating a word with the empty word is the original word.

In some applications, especially in logic, the alphabet is also known as the vocabulary and words are known as formulas or sentences; this breaks the letter/word metaphor and replaces it by a word/sentence metaphor.

Definition

A formal language L over an alphabet Σ is a subset of Σ^* , that is, a set of words over that alphabet. Sometimes the sets of words are grouped into expressions, whereas rules and constraints may be formulated for the creation of 'well-formed expressions'.

In computer science and mathematics, which do not usually deal with natural languages, the adjective "formal" is often omitted as redundant.

While formal language theory usually concerns itself with formal languages that are described by some syntactical rules, the actual definition of the concept "formal language" is only as above: a (possibly infinite) set of finite-length strings composed from a given alphabet, no more and no less. In practice, there are many languages that can be described by rules, such as regular languages or context-free languages. The notion of a formal grammar may be closer to the intuitive concept of a "language," one described by syntactic rules. By an abuse of the definition, a particular formal language is often thought of as being equipped with a formal grammar that describes it.

Kleene Closure

In mathematical logic and computer science, the Kleene star (or Kleene operator or Kleene closure) is a unary operation, either on sets of strings or on sets of symbols or characters. In mathematics it is more commonly known as the free monoid construction. The application of the Kleene star to a set V is written as V^* . It is widely used for regular expressions, which is the context in which it was introduced by Stephen Kleene to characterize certain automata, where it means "zero or more".

1. If V is a set of strings, then V^* is defined as the smallest superset of V that contains the empty string ϵ and is closed under the string concatenation operation.
2. If V is a set of symbols or characters, then V^* is the set of all strings over symbols in V , including the empty string ϵ .

The set V^* can also be described as the set of finite-length strings that can be generated by concatenating arbitrary elements of V , allowing the use of the same element multiple times. If V is either the empty set \emptyset or the singleton set $\{\epsilon\}$, then $V^* = \{\epsilon\}$; if V is any other finite set or countably infinite set, then V^* is a countably infinite set.^[1] As a consequence, each formal language over a finite or countably infinite alphabet is countable.

The operators are used in rewrite rules for generative grammars.

Definition and notation

Given a set V define

$V^0 = \{\epsilon\}$ (the language consisting only of the empty string),

$V^1 = V$

and define recursively the set

$V^{i+1} = \{wv : w \in V^i \text{ and } v \in V\}$ for each $i > 0$.

If V is a formal language, then V^i , the i -th power of the set V , is a shorthand for the concatenation of set V with itself i times. That is, V^i can be understood to be the set of all strings that can be represented as the concatenation of i strings in V .

The definition of Kleene star on V is^[2]

This means that the Kleene star operator is an idempotent unary operator: $(V^*)^* = V^*$ for any set V of strings or characters, as $(V^*)^i = V^*$ for every $i \geq 1$.

Arithmetic expressions

An arithmetic expression is an expression that results in a numeric value. There are two kinds of numeric values, integers (whole numbers), and real or floating point numbers (numbers containing a decimal point).

The simplest arithmetic expressions are literals (the number itself, written with digits) and variables (named values):

Example	Description
12	A literal integer, representing the number 12.
-5	A literal integer, representing the number negative 5.
-5.0	A literal real, representing the number negative 5.
5.0E4	A literal real, representing the number 50000.

count	A variable. If it was declared as <code>int count</code> , it will hold an integer value; but if declared as <code>double count</code> , it will hold a real value.
-------	---

More complex arithmetic expressions can be formed by connecting literals and variables with one of the arithmetic operators:

Operator	Meaning
+	Add.
-	Subtract.
*	Multiply (it's difficult to type the usual multiplication symbol).
/	Divide (it's difficult to type the usual division symbol). Division of two integer values will give an integer result (any fractional part is just discarded). For example, <code>14/5</code> gives 2. This is called integer division.
%	Mod (remainder). Used for integers only, this operation gives the remainder of a division; for example, <code>14%5</code> gives 4. The sign (positive or negative) of the result is the same as the sign of the first number.

Parentheses may be used to control the order in which the operators are applied. If you don't use parentheses, operations with higher precedence are done first.

Notes:

- If you mix integers and reals in an operation, the result is a real. For example, `3*5.0` is 15.0, not 15.
- You can assign an integer value to a real variable. For example, `double x=5` sets `x` to 5.0.
- You cannot assign a real value to an integer variable. For example, both `int i=5.8` and `int i=5.0` are illegal. This is to protect you from accidentally

losing the fractional part. However, if you use a cast to reassure Java that you really mean it, then it's legal. For example, `int i=(int)5.8` is legal, and gives `i` the value 5.

- By far the most commonly used numeric types are `int` and `double`. However, there are other numeric primitive types.

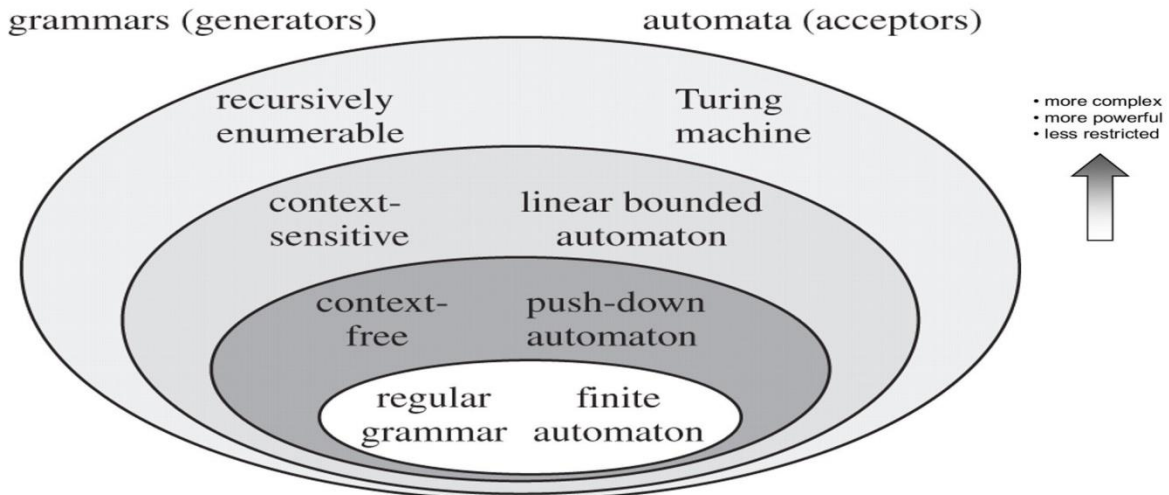
Chomsky Hierarchy

Any language is a structured medium of communication whether it is a spoken or written natural language, sign or coded language, or a formal programming language. Languages are characterised by two basic elements – syntax (grammatical rules) and semantics (meaning). In some languages, the meaning might vary depending upon a third factor called context of usage.

Depending on restrictions and complexity present in the grammar, languages find a place in the hierarchy of formal languages. Noam Chomsky, celebrated American linguist cum cognitive scientist, defined this hierarchy in 1956 and hence it's called Chomsky Hierarchy.

Although his concept is quite old, there's renewed interest because of its relevance to Natural Language Processing. Chomsky hierarchy helps us answer questions like “Can a natural language like English be described (‘parsed’, ‘compiled’) with the same methods as used for formal/artificial (programming) languages in computer science?”

Discussion



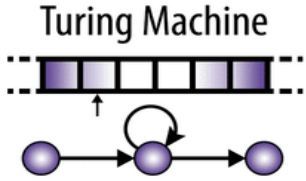

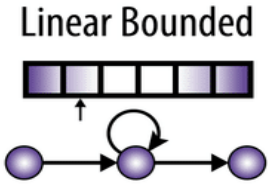

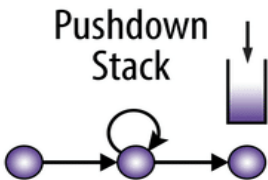
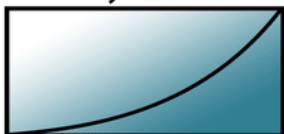
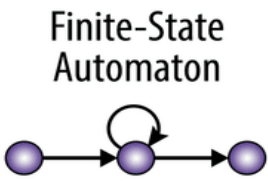

- There are 4 levels – Type-3, Type-2, Type-1, Type-0. With every level, the grammar becomes less restrictive in rules, but more complicated to automate. Every level is also a subset of the subsequent level.
- Type-3: Regular Grammar - most restrictive of the set, they generate regular languages. They must have a single non-terminal on the left-hand-side and a right-hand-side consisting of a single terminal or single terminal followed by a single non-terminal.
- Type-2: Context-Free Grammar - generate context-free languages, a category of immense interest to NLP practitioners. Here all rules take the form $A \rightarrow \beta$, where A is a single non-terminal symbol and β is a string of symbols.
- Type-1: Context-Sensitive Grammar - the highest programmable level, they generate context-sensitive languages. They have rules of the form $\alpha A \beta \rightarrow \alpha \gamma \beta$ with A as a non-terminal and α, β, γ as strings of terminals and non-terminals. Strings α, β may be empty, but γ must be nonempty.
- Type-0: Recursively enumerable grammar - are too generic and unrestricted to describe the syntax of either programming or natural languages.

What are the common terms and definitions used while studying Chomsky Hierarchy?

- Symbol - Letters, digits, single characters. Example - A,b,3
- String - Finite sequence of symbols. Example - Abcd, x12
- Production Rules - Set of rules for every grammar describing how to form strings from the language that are syntactically valid.
- Terminal - Smallest unit of a grammar that appears in production rules, cannot be further broken down.
- Non-terminal - Symbols that can be replaced by other non-terminals or terminals by successive application of production rules.
- Grammar - Rules for forming well-structured sentences and the words that make up those sentences in a language. A 4-tuple $G = (V, T, P, S)$ such that $V =$ Finite non-

empty set of non-terminal symbols, T = Finite set of terminal symbols, P = Finite non-empty set of production rules, S = Start symbol

- Language - Set of strings conforming to a grammar. Programming languages have finite strings, most natural languages are seemingly infinite. Example – Spanish, Python, Hexadecimal code.
- Automaton - Programmable version of a grammar governed by pre-defined production rules. It has clearly set computing requirements of memory and processing. Example – Regular automaton for regex.

Language	Automaton	Grammar	Recognition
Recursively Enumerable Languages	<p>Turing Machine</p> 	<p>Unrestricted</p> $Baa \rightarrow A$	<p>Undecidable</p> 
Context-Sensitive Languages	<p>Linear Bounded</p> 	<p>Context Sensitive</p> $A t \rightarrow aA$	<p>Exponential?</p> 
Context-Free Languages	<p>Pushdown Stack</p> 	<p>Context Free</p> $S \rightarrow gS c$	<p>Polynomial</p> 
Regular Languages	<p>Finite-State Automaton</p> 	<p>Regular</p> $A \rightarrow cA$	<p>Linear</p> 

- Under Type-3 grammar, we don't classify entire languages as the production rules are restrictive. However, constructs inside a language describable by regular expressions come under this type.

For instance, rule for naming an identifier in a programming language – regular expression with any combination of case-insensitive letters, some special characters and numbers, but must start with a letter.

Context-free languages classified as Type-2 are capable of handling an important language construct called nested dependencies. English example – Recursive presence of “If <phrase> then <phrase>” – “If it rains today and if I don't carry an umbrella, then I'd get drenched”. For programming languages, the matching parentheses of functions or loops get covered by this grammar.

In Type-1 languages, placing the restriction on productions $\alpha \rightarrow \beta$ of a phrase structure that β be at least as long as α , they become context sensitive. They permit replacement of α by β only in a 'context', $[\text{context}] \alpha [\text{context}] \rightarrow [\text{context}] \beta [\text{context}]$.

Finally, Type-0 languages have no restrictions on their grammar and may loop forever. They don't have an algorithm enumerating all the elements.

- What are the type of Automaton that recognizes the grammar in each level?
- Type-3: Finite-State Automata - To compute constructs for a regular language, the most important consideration is that there is no memory requirement. Think of a single purpose vending machine for platform tickets or a lift algorithm. The automaton knows the present state and next permissible states, but does not 'remember' past steps.
- Type-2: Push-Down Automata - In order to match nested dependencies, this automaton requires a one-ended memory stack. For instance, to match the number of 'if' and 'else' phrases, the automaton needs to 'remember' the latest occurring 'if'. Only then it can find the corresponding 'else'.

- Type-1: Linear-Bounded Automata - is a form of a restricted Turing machine which instead of being unlimited, is bounded by some computable linear function. The advantage of this automaton is that its memory requirement (RAM upper limit) is predictable even if the execution is recursive in parts.
- Type-0: Turing Machine - Non-computable functions exist in Mathematics and Computer Science. The Turing machine however allows representing even such functions as a sequence of discrete steps. Control is finite even if data might be seemingly infinite.

Can you give a quick example for each type of grammar/language?

- Type-3: Regex to define tokens such as identifiers, language keywords in programming languages. A coin vending machine that accepts only 1-Rupee, 2-Rupee and 5-Rupee coins has a regular language with only three words – 1, 2, 5.
- Type-2: Statement blocks in programming languages such as functions in parentheses, If-Else, for loops. In natural language, nouns and their plurals can be recognized through one NFA, verbs and their different forms can be recognized through another NFA, and then combined. Singular (The girl runs home → Girl + Runs). Plural (The girls run home → Girls + Run)
- Type-1: Though most language constructs in natural language are context-free, in some situations linear matching of tokens has to be done, such as - "The square roots of 16, 9 and 4 are 4, 3 and 2, respectively." Here 16 is to be matched with 4, 9 is matched with 3, and 4 is matched with 2.
- Type-0: A language with no restrictions is not conducive to communication or automation. Hence there are no common examples for this type. However, some mathematical seemingly unsolvable equations are expressed in this form.

In which level of the hierarchy do formal programming languages fall?

Reading a text file containing a high-level language program and compiling it as per its syntax is done in two steps.

Finite state models associated with Type-3 grammar are used for performing the first step of lexical analysis. Raw text is aggregated into keywords, strings, numerical constants and identifiers in this step.

In the second step, to parse the program constructs of any high level language according to its syntax, a Context-Free Grammar is required. Usually these grammars are specified in Backus-Naur Form (BNF).

For example, to build a grammar for IF statement, grammar would begin with a non-terminal statement S. Rules will be of the form:

$S \rightarrow \text{IF-STATEMENT}$

$\text{IF-STATEMENT} \rightarrow \text{if CONDITION then BLOCK endif}$

$\text{BLOCK} \rightarrow \text{STATEMENT} \mid \text{BLOCK};$

Conventionally, all high-level programming languages can be covered under the Type-2 grammar in Chomsky's hierarchy.

Python language has a unique feature of being white-space sensitive. To make this feature fit into a conventional CFG, Python uses two additional tokens 'INDENT' and 'DEDENT' to represent the line indentations.

However, just syntactic analysis does not guarantee that the language will be entirely 'understood'. Semantics need to match too.

Where can we place natural languages in the hierarchy?

Natural languages are an infinite set of sentences constructed out of a finite set of characters. Words in a sentence don't have defined upper limits either. When natural languages are reverse engineered into their component parts, they get broken down into 4 parts - syntax, semantics, morphology, phonology.

Tokenising words and identifying nested dependencies work as explained in the previous section.

Part-of-Speech Tagging is a challenge. “He runs 20 miles every day” and “The batsman scored 150 runs in one day” – the same word ‘runs’ becomes a noun and verb. Finite state grammars can be used for resolving such lexical ambiguity.

Identifying cases (subjective - I, possessive - Mine, objective - Me, etc) for nouns varies across languages. Old English has 5, Modern English – 3, Sanskrit and Tamil - clearly defined 8 cases. Each case also has interrogative forms. Clear definition of cases enables free word order. The CFG defined for these languages take care of this.

Natural languages are believed to be at least context-free. However, Dutch and Swiss German contain grammatical constructions with cross-serial dependencies which make them context sensitive.

Languages having clear and singular source text of grammar are easier to classify.

Are there any exceptional cases in natural languages that make its classification ambiguous?

NLP practitioners have successfully managed to assign a majority of natural language aspects to the regular and CFG category. However, some aspects don't easily conform to a particular grammar and require special handling.

- Structural ambiguity – Example ‘I saw the man with the telescope’. A CFG can assign two or more phrase structures (“parse trees”) to one and the same sequence of terminal symbols (words or word classes).
- Ungrammatical speech – Humans often talk in sentences that are incorrect grammatically. Missing words sometimes are implied in a sentence, not uttered explicitly. So decoding such sentences is a huge challenge as they don't qualify as per any defined grammar, but a native speaker can easily understand them.

- Sarcasm or proverb usage – When we say something but mean something entirely different. Here the semantic analysis becomes critical. We don't build grammars for these cases, we just prepare an exhaustive reference data set.
- Mixed language use – Humans often mix words from multiple languages. So computing systems need to identify all the constituent language words present in the sentence and then assign them to their respective grammars.

Regular expressions

A Regular Expression can be recursively defined as follows –

- ϵ is a Regular Expression indicates the language containing an empty string. ($L(\epsilon) = \{\epsilon\}$)
- ϕ is a Regular Expression denoting an empty language. ($L(\phi) = \{\}$)
- x is a Regular Expression where $L = \{x\}$
- If X is a Regular Expression denoting the language $L(X)$ and Y is a Regular Expression denoting the language $L(Y)$, then
 - $X + Y$ is a Regular Expression corresponding to the language $L(X) \cup L(Y)$ where $L(X+Y) = L(X) \cup L(Y)$.
 - $X . Y$ is a Regular Expression corresponding to the language $L(X) . L(Y)$ where $L(X.Y) = L(X) . L(Y)$
 - R^* is a Regular Expression corresponding to the language $L(R^*)$ where $L(R^*) = (L(R))^*$
- If we apply any of the rules several times from 1 to 5, they are Regular Expressions.

Some RE Examples

Regular Expressions	Regular Set
$(0 + 10^*)$	$L = \{0, 1, 10, 100, 1000, 10000, \dots\}$
(0^*10^*)	$L = \{1, 01, 10, 010, 0010, \dots\}$

$(0 + \epsilon)(1 + \epsilon)$	$L = \{\epsilon, 0, 1, 01\}$
$(a+b)^*$	Set of strings of a's and b's of any length including the null string. So $L = \{\epsilon, a, b, aa, ab, bb, ba, aaa, \dots\}$
$(a+b)^*abb$	Set of strings of a's and b's ending with the string abb. So $L = \{abb, aabb, babb, aaabb, ababb, \dots\}$
$(11)^*$	Set consisting of even number of 1's including empty string, So $L = \{\epsilon, 11, 1111, 111111, \dots\}$
$(aa)^*(bb)^*b$	Set of strings consisting of even number of a's followed by odd number of b's, so $L = \{b, aab, aabbb, aabbbb, aaaab, aaaabbb, \dots\}$
$(aa + ab + ba + bb)^*$	String of a's and b's of even length can be obtained by concatenating any combination of the strings aa, ab, ba and bb including null, so $L = \{aa, ab, ba, bb, aaab, aaba, \dots\}$

Generalized Transition graph

There are two methods to convert FA to regular expression –

1. State Elimination Method –

- **Step 1 –**

If the start state is an accepting state or has transitions in, add a new non-accepting start state and add an ϵ -transition between the new start state and the former start state.

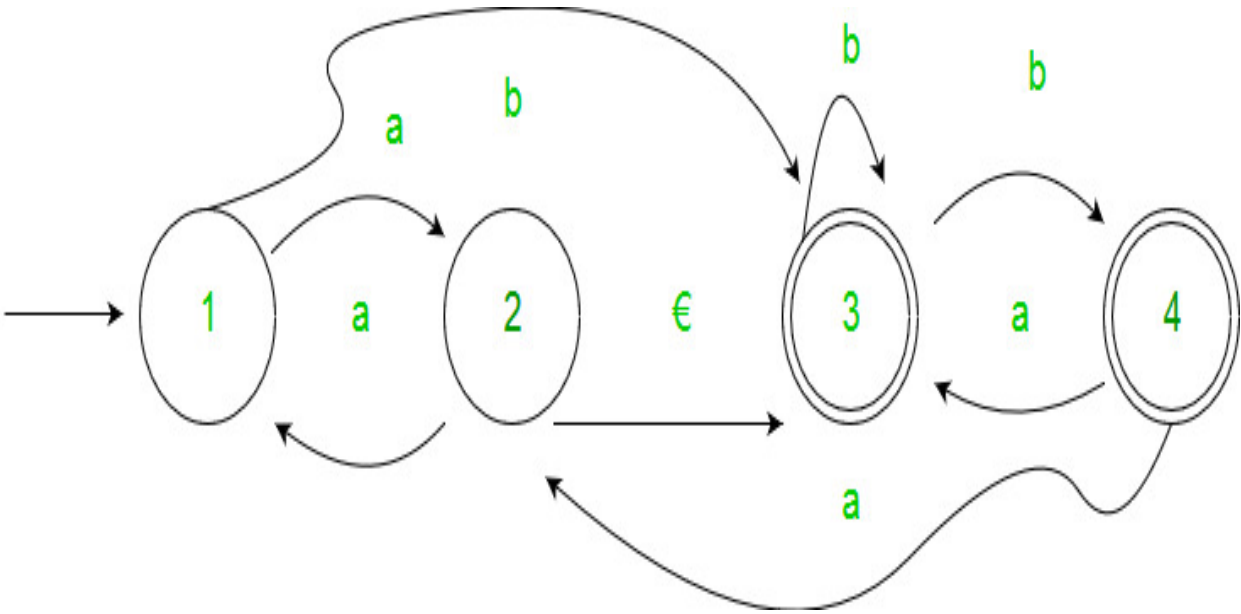
- **Step 2 –**

If there is more than one accepting state or if the single accepting state has transitions out, add a new accepting state, make all other states non-accepting, and add an ϵ -transition from each former accepting state to the new accepting state.

- **Step 3 –**

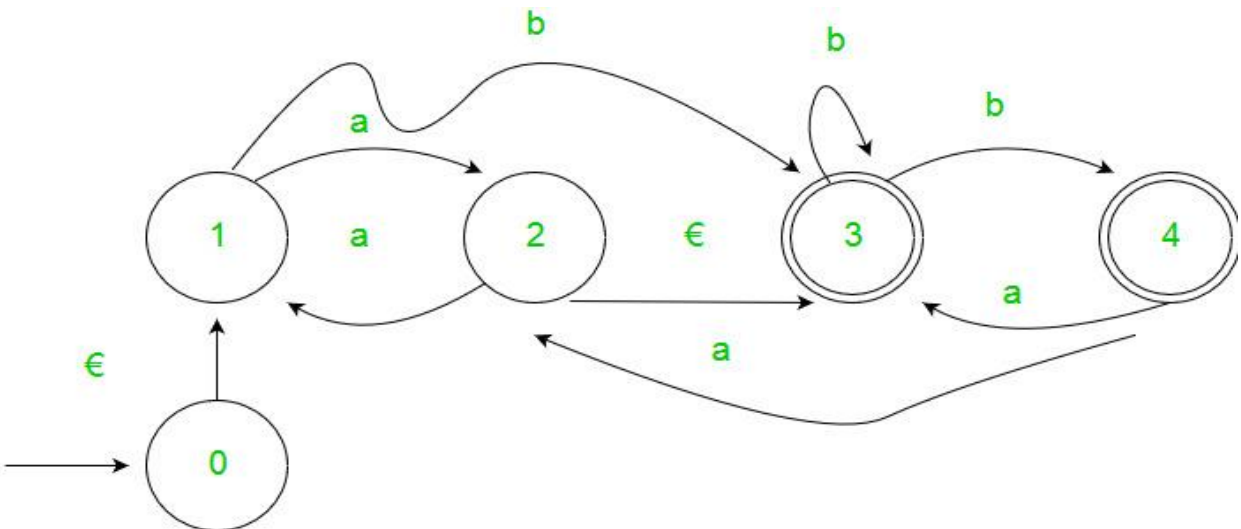
For each non-start non-accepting state in turn, eliminate the state and update transitions accordingly.

Example :-

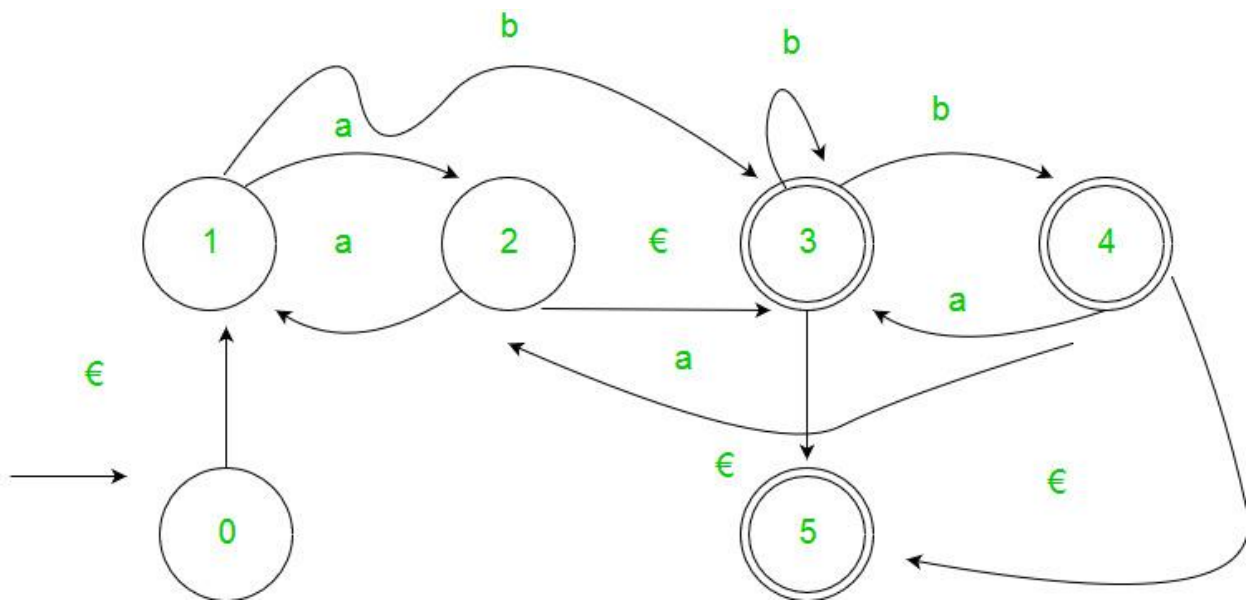


Solution :-

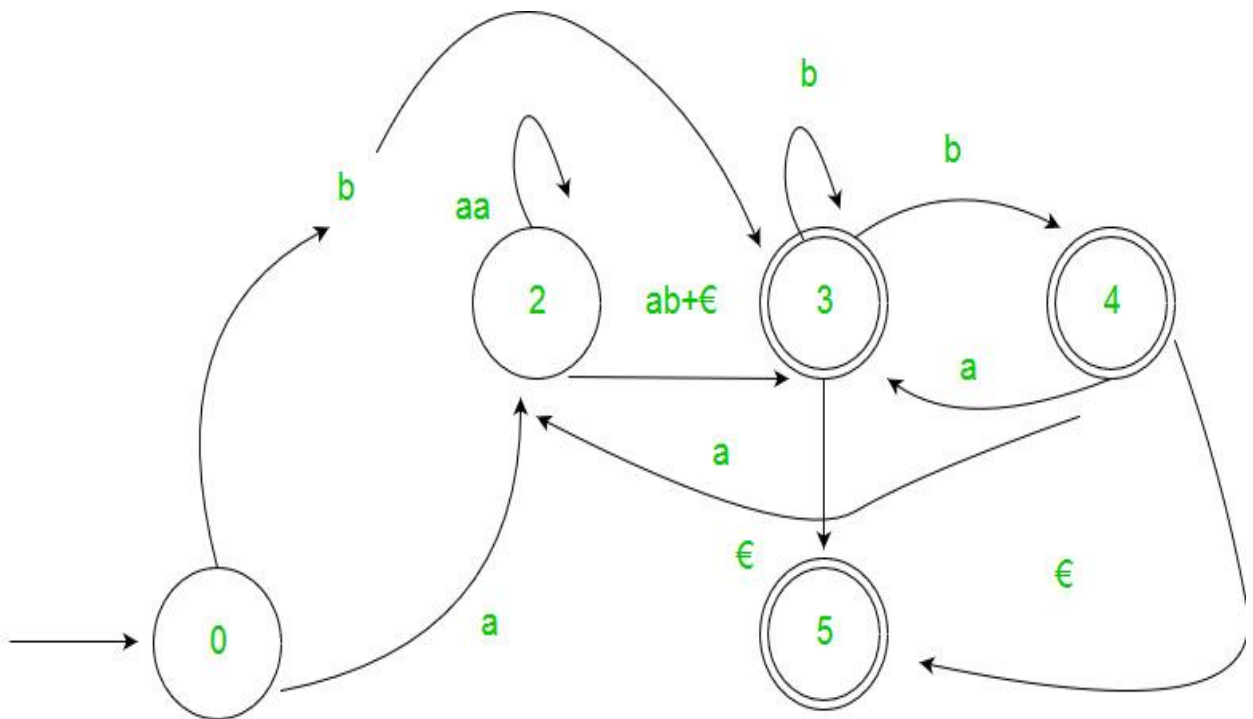
Step 1

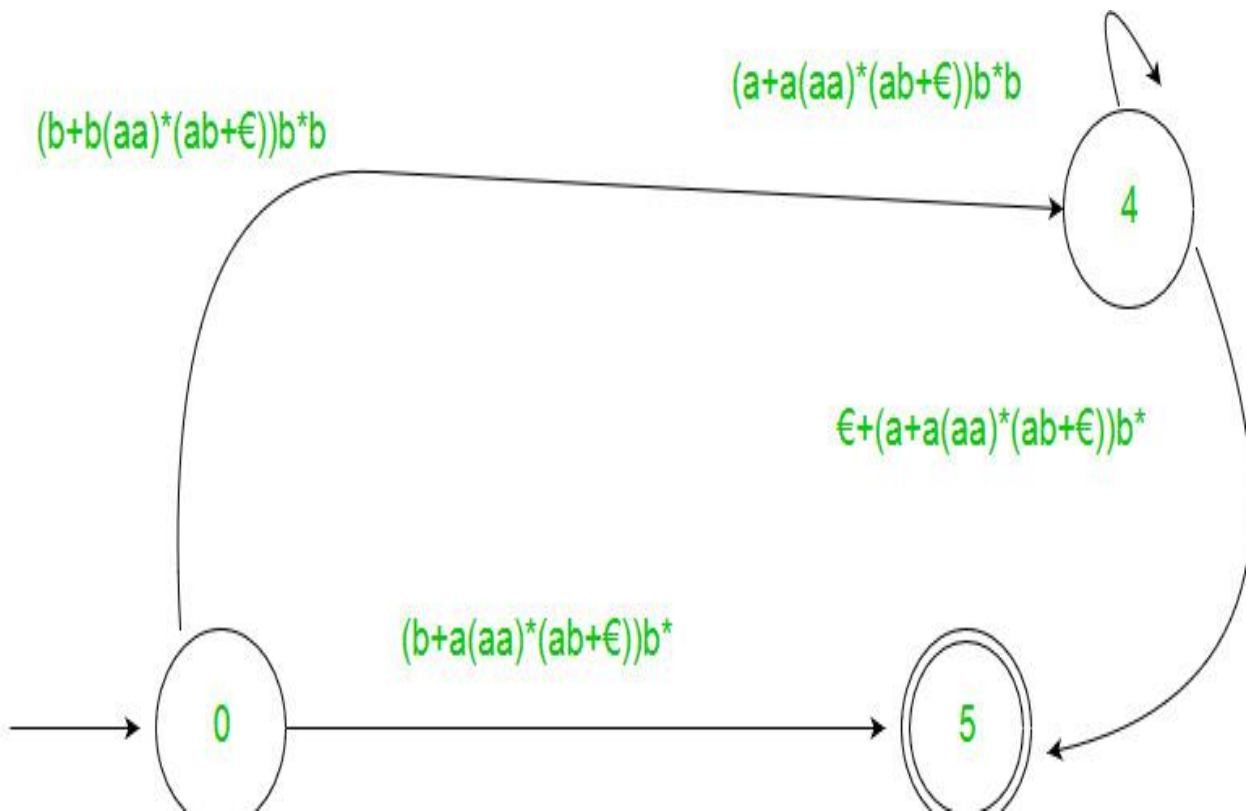
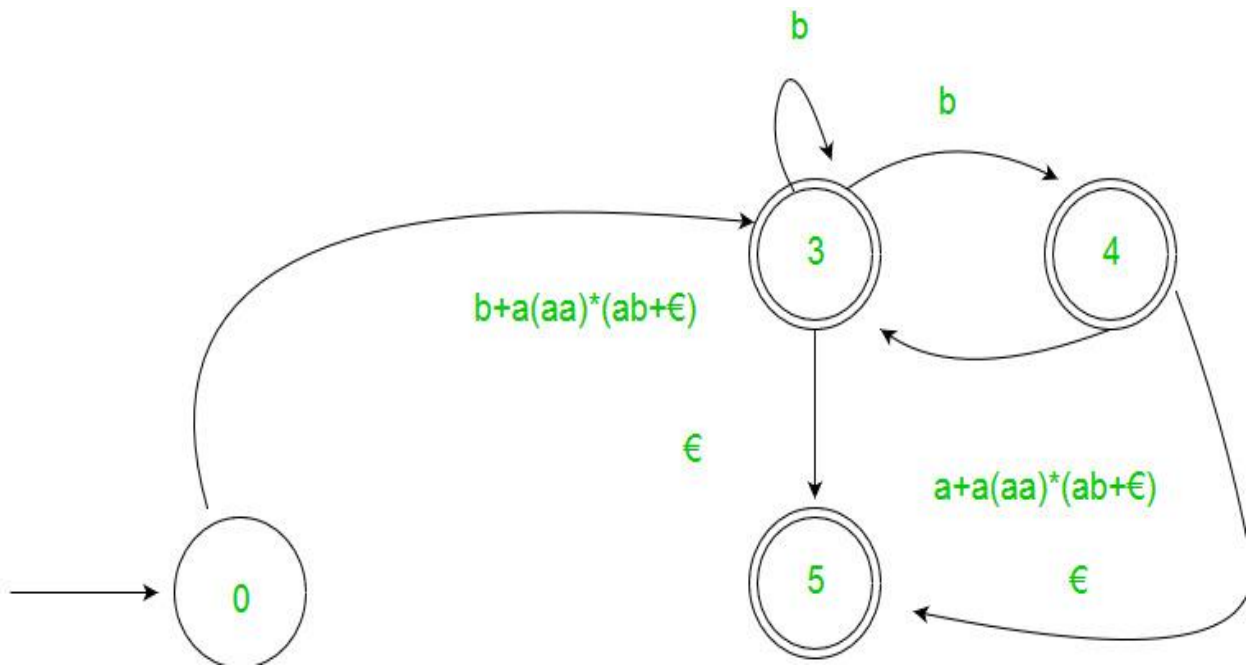


Step 2



Step 3





$$(b+a((aa)^*(ab+\epsilon))b^*+((b+a(aa)^*(ab+\epsilon))b^*b((a+a(aa)^*(ab+\epsilon))b^*b)^*(\epsilon+(a+a(aa)^*(ab+\epsilon))b^*))$$



2. Arden's Theorem – Let P and Q be 2 regular expressions. If P does not contain null string, then following equation in R, viz $R = Q + RP$, Has a unique solution by $R = QP^*$

Assumptions –

- The transition diagram should not have ϵ -moves.
- It must have only one initial state.

Using Arden's Theorem to find Regular Expression of Deterministic Finite automata –

1. For getting the regular expression for the automata we first create equations of the given form for all the states

$$q_1 = q_1w_{11} + q_2w_{21} + \dots + q_nw_{n1} + \epsilon \text{ (} q_1 \text{ is the initial state)}$$

$$q_2 = q_1w_{12} + q_2w_{22} + \dots + q_nw_{n2}$$

.

.

.

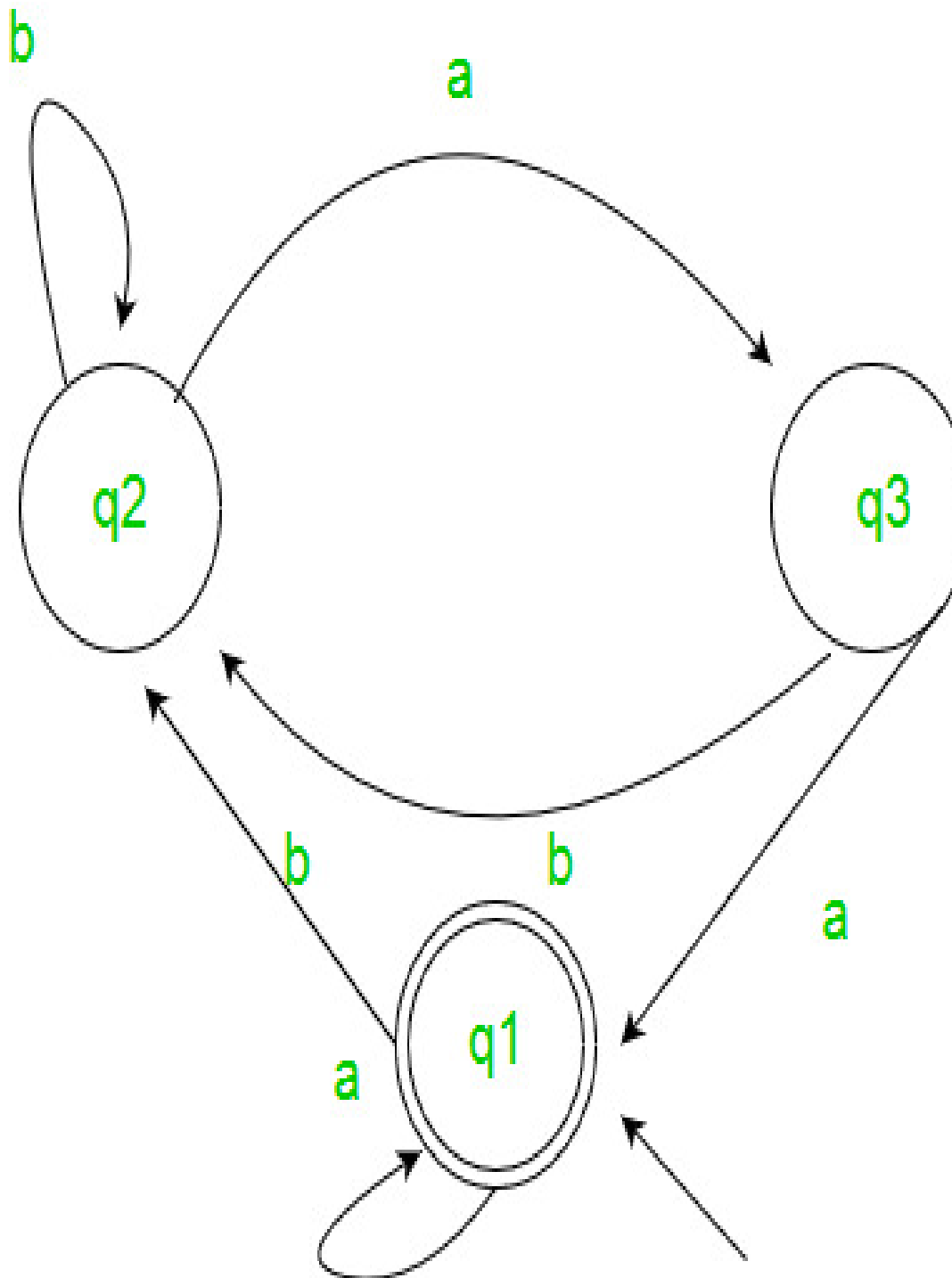
$$q_n = q_1w_{1n} + q_2w_{2n} + \dots + q_nw_{nn}$$

w_{ij} is the regular expression representing the set of labels of edges from q_i to q_j

Note – For parallel edges there will be that many expressions for that state in the expression.

2. Then we solve these equations to get the equation for q_i in terms of w_{ij} and that expression is the required solution, where q_i is a final state.

Example :-



Solution :-

Here the initial state is q_2 and the final state is q_1 .

The equations for the three states q_1 , q_2 , and q_3 are as follows ?

$$q_1 = q_1a + q_3a + \epsilon \quad (\epsilon \text{ move is because } q_1 \text{ is the initial state})$$

$$q_2 = q_1b + q_2b + q_3b$$

$$q_3 = q_2a$$

Now, we will solve these three equations ?

$$q_2 = q_1b + q_2b + q_3b$$

$$= q_1b + q_2b + (q_2a)b \quad (\text{Substituting value of } q_3)$$

$$= q_1b + q_2(b + ab)$$

$$= q_1b (b + ab)^* \quad (\text{Applying Arden's Theorem})$$

$$q_1 = q_1a + q_3a + \epsilon$$

$$= q_1a + q_2aa + \epsilon \quad (\text{Substituting value of } q_3)$$

$$= q_1a + q_1b(b + ab^*)aa + \epsilon \quad (\text{Substituting value of } q_2)$$

$$= q_1(a + b(b + ab)^*aa) + \epsilon$$

$$= \epsilon (a + b(b + ab)^*aa)^*$$

$$= (a + b(b + ab)^*aa)^*$$

Hence, the regular expression is $(a + b(b + ab)^*aa)^*$.

GATE CS Corner Questions

Practicing the following questions will help you test your knowledge. All questions have been asked in GATE in previous years or in GATE Mock Tests. It is highly recommended that you practice them.

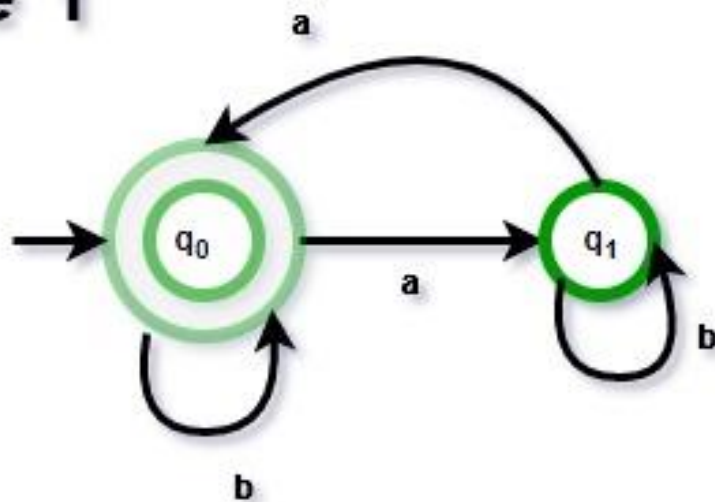
UNIT-IV

Conversion of regular expression to Finite Automata

In this article, we will see some popular regular expressions and how we can convert them to finite automata.

- **Even number of a's** : The regular expression for even number of a's is $(b|ab^*ab^*)^*$. We can construct a finite automata as shown in Figure 1.

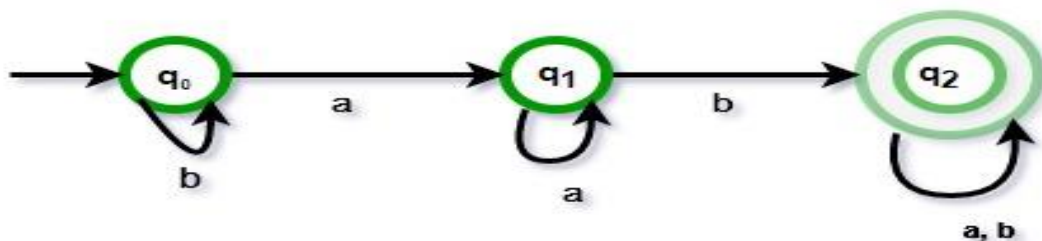
Figure 1



The above automata will accept all strings which have even number of a's. For zero a's, it will be in q_0 which is final state. For one 'a', it will go from q_0 to q_1 and the string will not be accepted. For two a's at any positions, it will go from q_0 to q_1 for 1st 'a' and q_1 to q_0 for second 'a'. So, it will accept all strings with even number of a's.

- **String with 'ab' as substring** : The regular expression for strings with 'ab' as substring is $(a|b)^*ab(a|b)^*$. We can construct finite automata as shown in Figure 2.

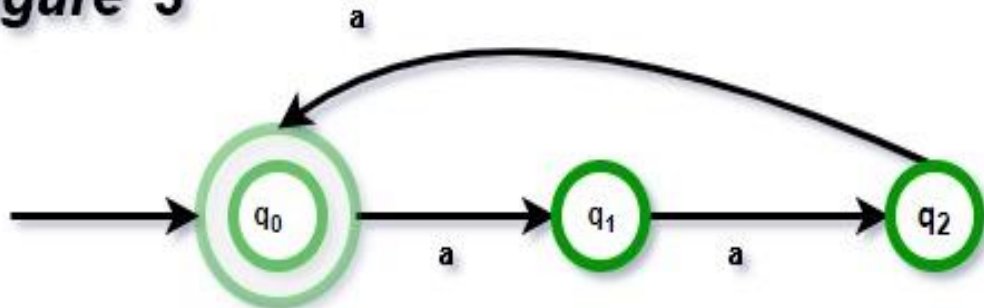
Figure 2



The above automata will accept all string which have 'ab' as substring. The automata will remain in initial state q_0 for b's. It will move to q_1 after reading 'a' and remain in same state for all 'a' afterwards. Then it will move to q_2 if 'b' is read. That means, the string has read 'ab' as substring if it reaches q_2 .

String with count of 'a' divisible by 3 : The regular expression for strings with count of a divisible by 3 is $\{a^{3n} \mid n \geq 0\}$. We can construct automata as shown in Figure 3.

Figure 3



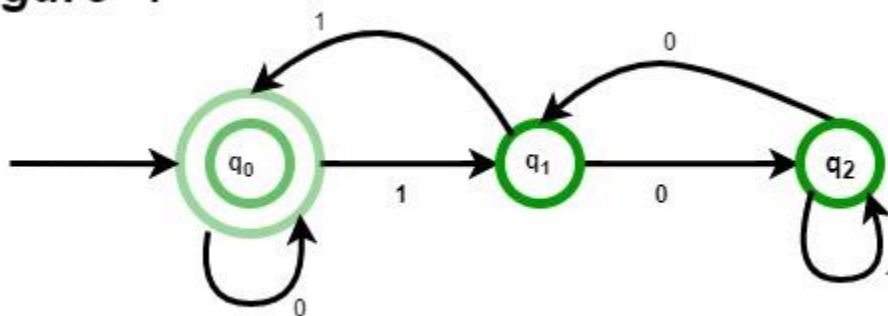
The above automata will accept all string of form a^{3n} . The automata will remain in initial state q_0 for ϵ and it will be accepted. For string 'aaa', it will move from q_0 to q_1 then q_1 to q_2 and then q_2 to q_0 . For every set of three a's, it will come to q_0 , hence accepted. Otherwise, it will be in q_1 or q_2 , hence rejected.

Note : If we want to design a finite automata with number of a's as $3n+1$, same automata can be used with final state as q_1 instead of q_0 .

If we want to design a finite automata with language $\{a^{kn} \mid n \geq 0\}$, k states are required. We have used $k = 3$ in our example.

Binary numbers divisible by 3 : The regular expression for binary numbers which are divisible by three is $(0|1(01^*0)^*1)^*$. The examples of binary number divisible by 3 are 0, 011, 110, 1001, 1100, 1111, 10010 etc. The DFA corresponding to binary number divisible by 3 can be shown in Figure 4.

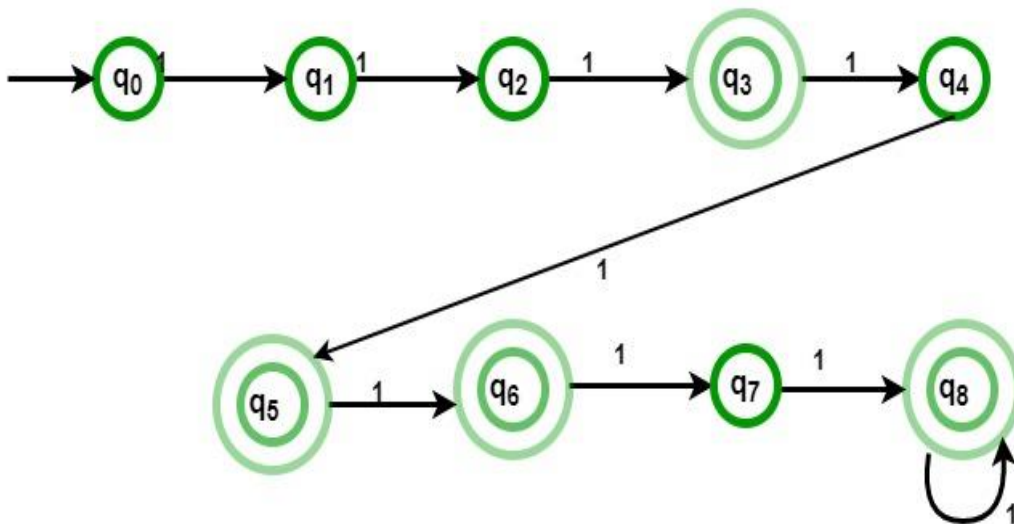
Figure 4



The above automata will accept all binary numbers divisible by 3. For 1001, the automata will go from q_0 to q_1 , then q_1 to q_2 , then q_2 to q_1 and finally q_2 to q_0 , hence accepted. For 0111, the automata will go from q_0 to q_0 , then q_0 to q_1 , then q_1 to q_0 and finally q_0 to q_1 , hence rejected.

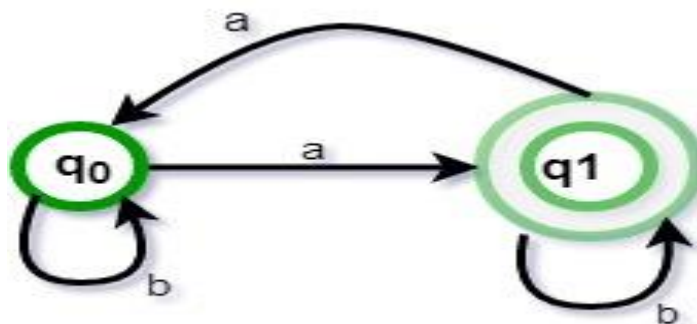
String with regular expression $(111 + 11111)^*$: The string accepted using this regular expression will have 3, 5, 6(111 twice), 8 (11111 once and 111 once), 9 (111 thrice), 10 (11111 twice) and all other counts of 1 afterwards. The DFA corresponding to given regular expression is given in Figure 5.

Figure 5



Question : What will be the minimum number of states for strings with odd number of a 's?

Solution : The regular expression for odd number of a is $b^*ab^*(ab^*ab^*)^*$ and corresponding automata is given in Figure 6 and minimum number of states are 2.



TOC | Designing Deterministic Finite Automata (Set 1)

This article has been contributed by Sonal Tuteja.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Attention reader! Don't stop learning now. Get hold of all the important CS Theory concepts for SDE interviews with the CS Theory Course at a student-friendly price and become industry ready.

NFA

- NFA stands for non-deterministic finite automata. It is easy to construct an NFA than DFA for a given regular language.
- The finite automata are called NFA when there exist many paths for specific input from the current state to the next state.
- Every NFA is not DFA, but each NFA can be translated into DFA.
- NFA is defined in the same way as DFA but with the following two exceptions, it contains multiple next states, and it contains ϵ transition.

In the following image, we can see that from state q_0 for input a , there are two next states q_1 and q_2 , similarly, from q_0 for input b , the next states are q_0 and q_1 . Thus it is not fixed or determined that with a particular input where to go next. Hence this FA is called non-deterministic finite automata.

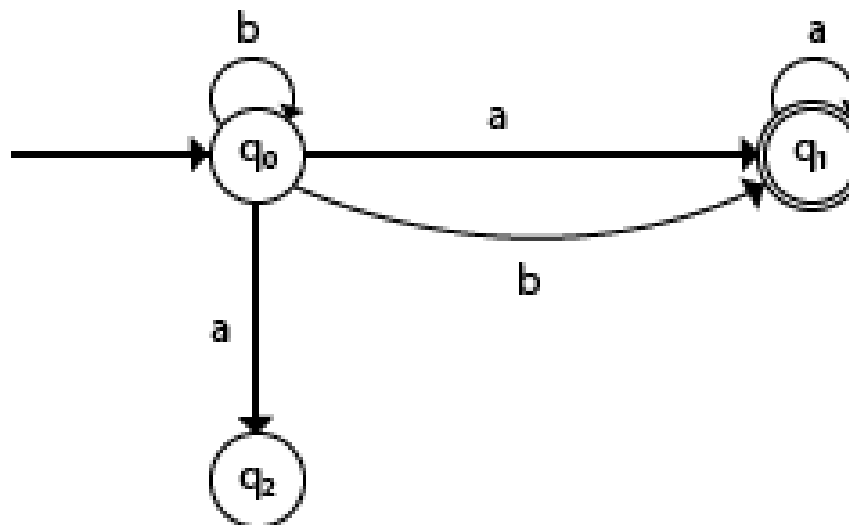


Fig:- NFA

Formal definition of NFA:

NFA also has five states same as DFA, but with different transition function, as shown follows:

$$\delta: Q \times \Sigma \rightarrow 2^Q$$

where,

1. Q : finite set of states
2. Σ : finite set of the input symbol
3. q_0 : initial state
4. F : final state
5. δ : Transition function

Graphical Representation of an NFA

An NFA can be represented by digraphs called state diagram. In which:

1. The state is represented by vertices.
2. The arc labeled with an input character show the transitions.
3. The initial state is marked with an arrow.
4. The final state is denoted by the double circle.

Example 1:

1. $Q = \{q_0, q_1, q_2\}$
2. $\Sigma = \{0, 1\}$
3. $q_0 = \{q_0\}$
4. $F = \{q_2\}$

Solution:

Transition diagram:

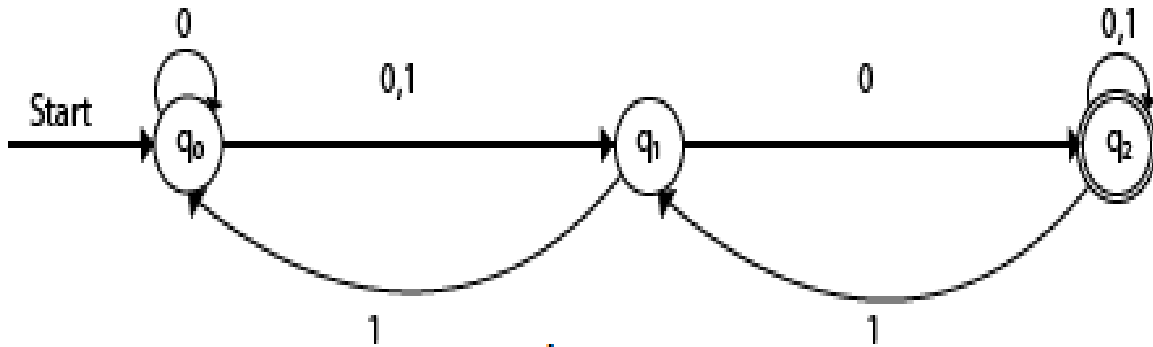


Fig: NFA

Transition Table:

Present State	Next state for Input 0	Next State of Input 1
→q0	q0, q1	q1
q1	q2	q0
*q2	q2	q1, q2

In the above diagram, we can see that when the current state is q0, on input 0, the next state will be q0 or q1, and on 1 input the next state will be q1. When the current state is q1, on input 0 the next state will be q2 and on 1 input, the next state will be q0. When the current state is q2, on 0 input the next state is q2, and on 1 input the next state will be q1 or q2.

Example 2:

NFA with $\Sigma = \{0, 1\}$ accepts all strings with 01.

Solution:

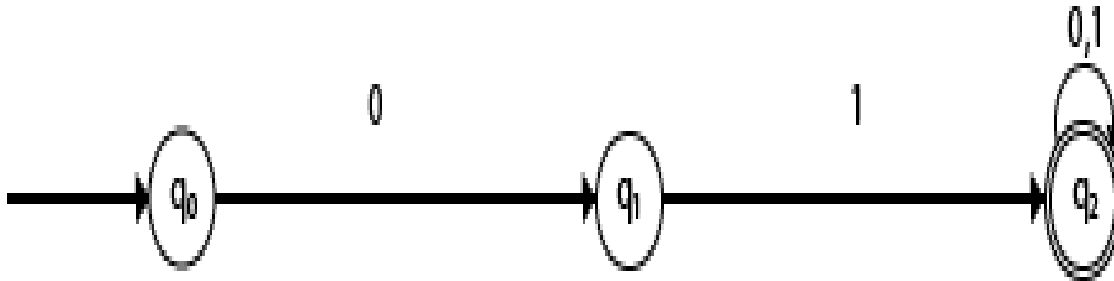


Fig: NFA

Transition Table:

Present State	Next state for Input 0	Next State of Input 1
→q0	q1	ε
q1	ε	q2
*q2	q2	q2

Example 3:

NFA with $\Sigma = \{0, 1\}$ and accept all string of length atleast 2.

Solution:

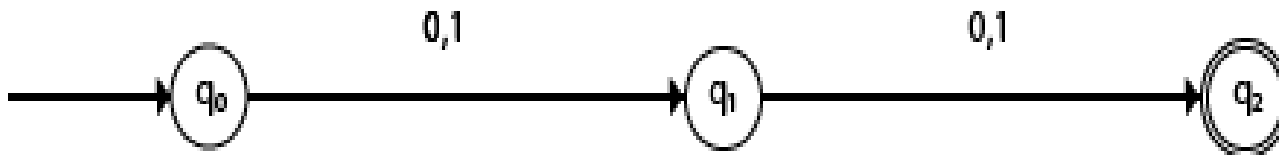


Fig: NFA

Transition Table:

Present State	Next state for Input 0	Next State of Input 1
$\rightarrow q_0$	q_1	q_1
q_1	q_2	q_2
$*q_2$	ϵ	ϵ

DFA

DFA (Deterministic finite automata)

- DFA refers to deterministic finite automata. Deterministic refers to the uniqueness of the computation. The finite automata are called deterministic finite automata if the machine is read an input string one symbol at a time.
- In DFA, there is only one path for specific input from the current state to the next state.
- DFA does not accept the null move, i.e., the DFA cannot change state without any input character.
- DFA can contain multiple final states. It is used in Lexical Analysis in Compiler.

In the following diagram, we can see that from state q_0 for input a , there is only one path which is going to q_1 . Similarly, from q_0 , there is only one path for input b going to q_2 .

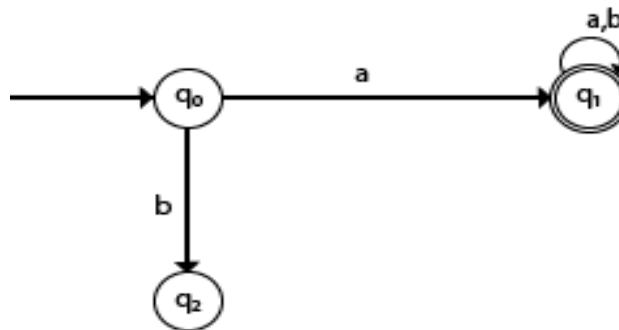


Fig:- DFA

Formal Definition of DFA

A DFA is a collection of 5-tuples same as we described in the definition of FA.

1. Q : finite set of states
2. Σ : finite set of the input symbol
3. q_0 : initial state
4. F : final state
5. δ : Transition function

Transition function can be defined as:

1. $\delta: Q \times \Sigma \rightarrow Q$

Graphical Representation of DFA

A DFA can be represented by digraphs called state diagram. In which:

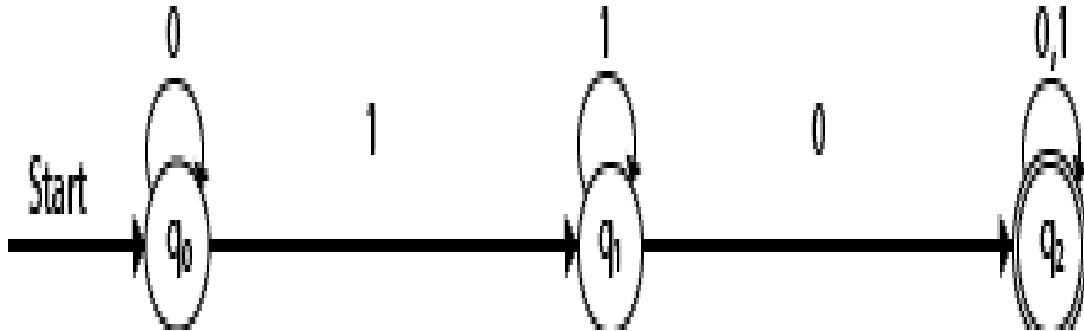
1. The state is represented by vertices.
2. The arc labeled with an input character show the transitions.
3. The initial state is marked with an arrow.
4. The final state is denoted by a double circle.

Example 1:

1. $Q = \{q_0, q_1, q_2\}$
2. $\Sigma = \{0, 1\}$
3. $q_0 = \{q_0\}$
4. $F = \{q_2\}$

Solution:

Transition Diagram:



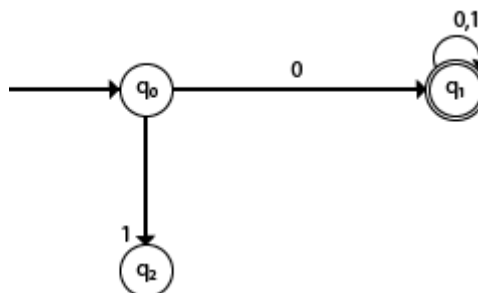
Transition Table:

Present State	Next state for Input 0	Next State of Input 1
→q0	q0	q1
q1	q2	q1
*q2	q2	q2

Example 2:

DFA with $\Sigma = \{0, 1\}$ accepts all starting with 0.

Solution:



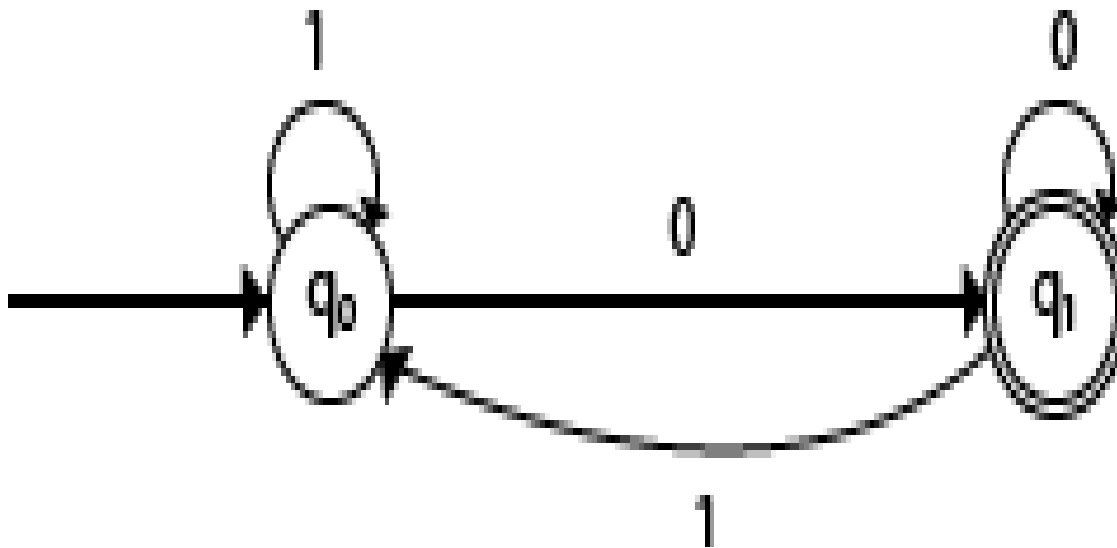
Explanation:

- In the above diagram, we can see that on given 0 as input to DFA in state q_0 the DFA changes state to q_1 and always go to final state q_1 on starting input 0. It can accept 00, 01, 000, 001....etc. It can't accept any string which starts with 1, because it will never go to final state on a string starting with 1.

Example 3:

DFA with $\Sigma = \{0, 1\}$ accepts all ending with 0.

Solution:



Explanation:

In the above diagram, we can see that on given 0 as input to DFA in state q_0 , the DFA changes state to q_1 . It can accept any string which ends with 0 like 00, 10, 110, 100....etc. It can't accept any string which ends with 1, because it will never go to the final state q_1 on 1 input, so the string ending with 1, will not be accepted or will be rejected.

Conversion of NFA to DFA

In this section, we will discuss the method of converting NFA to its equivalent DFA. In NFA, when a specific input is given to the current state, the machine goes to multiple states. It can have zero, one or more than one move on a given input symbol. On the

other hand, in DFA, when a specific input is given to the current state, the machine goes to only one state. DFA has only one move on a given input symbol.

Let, $M = (Q, \Sigma, \delta, q_0, F)$ is an NFA which accepts the language $L(M)$. There should be equivalent DFA denoted by $M' = (Q', \Sigma', q_0', \delta', F')$ such that $L(M) = L(M')$.

Steps for converting NFA to DFA:

Step 1: Initially $Q' = \phi$

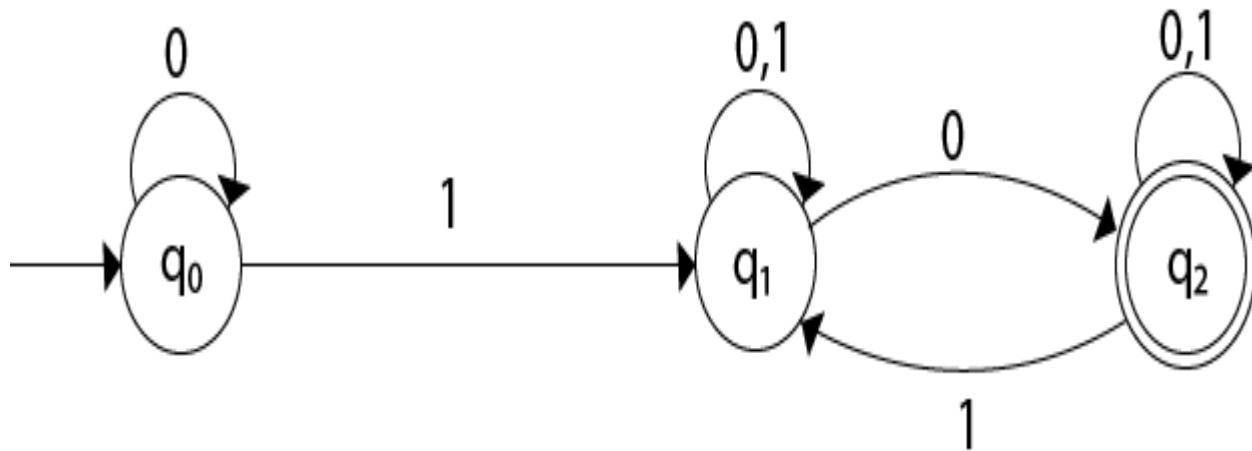
Step 2: Add q_0 of NFA to Q' . Then find the transitions from this start state.

Step 3: In Q' , find the possible set of states for each input symbol. If this set of states is not in Q' , then add it to Q' .

Step 4: In DFA, the final state will be all the states which contain F (final states of NFA)

Example 1:

Convert the given NFA to DFA.



Solution: For the given transition diagram we will first construct the transition table.

State	0	1
$\rightarrow q_0$	q0	q1
q1	{q1, q2}	q1

*q2	q2	{q1, q2}
-----	----	----------

Now we will obtain δ' transition for state q0.

1. $\delta'([q0], 0) = [q0]$
2. $\delta'([q0], 1) = [q1]$

The δ' transition for state q1 is obtained as:

1. $\delta'([q1], 0) = [q1, q2]$ (**new** state generated)
2. $\delta'([q1], 1) = [q1]$

The δ' transition for state q2 is obtained as:

1. $\delta'([q2], 0) = [q2]$
2. $\delta'([q2], 1) = [q1, q2]$

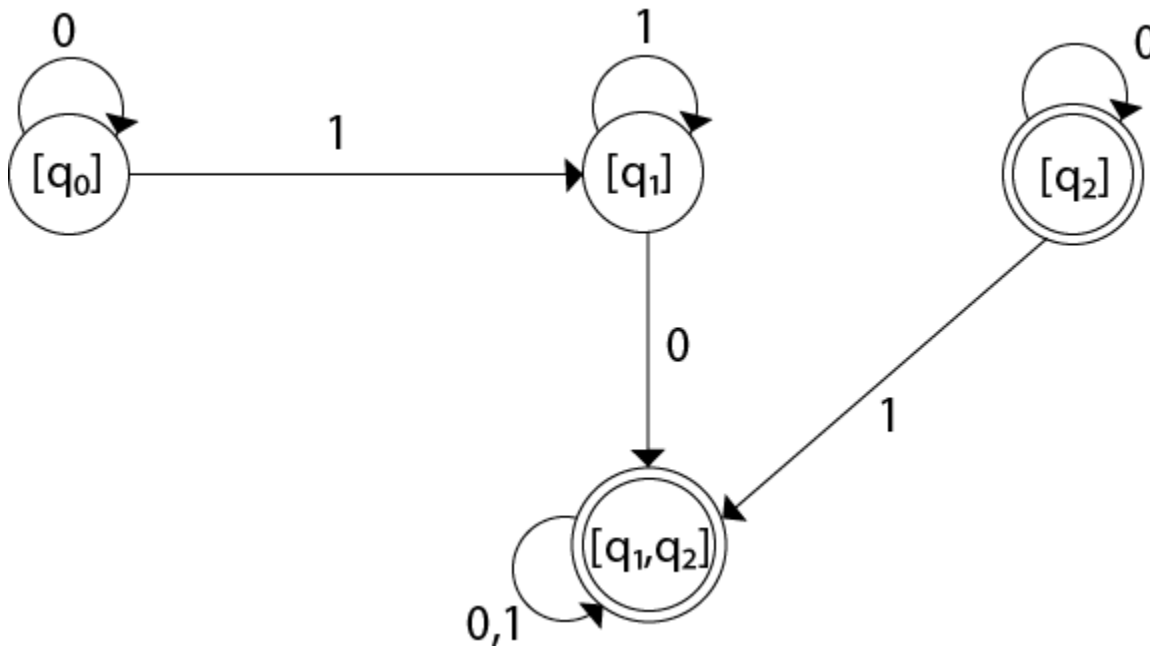
Now we will obtain δ' transition on [q1, q2].

1. $\delta'([q1, q2], 0) = \delta(q1, 0) \cup \delta(q2, 0)$
2. $= \{q1, q2\} \cup \{q2\}$
3. $= [q1, q2]$
4. $\delta'([q1, q2], 1) = \delta(q1, 1) \cup \delta(q2, 1)$
5. $= \{q1\} \cup \{q1, q2\}$
6. $= \{q1, q2\}$
7. $= [q1, q2]$

The state [q1, q2] is the final state as well because it contains a final state q2. The transition table for the constructed DFA will be:

State	0	1
$\rightarrow[q0]$	[q0]	[q1]
[q1]	[q1, q2]	[q1]
*[q2]	[q2]	[q1, q2]
*[q1, q2]	[q1, q2]	[q1, q2]

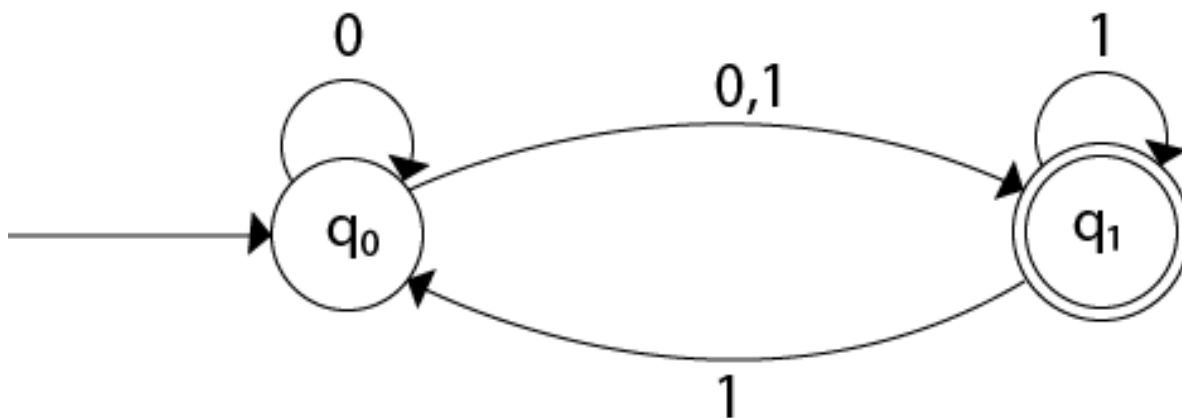
The Transition diagram will be:



The state q_2 can be eliminated because q_2 is an unreachable state.

Example 2:

Convert the given NFA to DFA.



Solution: For the given transition diagram we will first construct the transition table.

State	0	1
$\rightarrow q_0$	{q ₀ , q ₁ }	{q ₁ }
*q ₁	ϕ	{q ₀ , q ₁ }

Now we will obtain δ' transition for state q₀.

1. $\delta'([q_0], 0) = \{q_0, q_1\}$
2. $\quad \quad \quad = [q_0, q_1]$ (new state generated)
3. $\delta'([q_0], 1) = \{q_1\} = [q_1]$

The δ' transition for state q₁ is obtained as:

1. $\delta'([q_1], 0) = \phi$
2. $\delta'([q_1], 1) = [q_0, q_1]$

Now we will obtain δ' transition on [q₀, q₁].

1. $\delta'([q_0, q_1], 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$
2. $\quad \quad \quad = \{q_0, q_1\} \cup \phi$
3. $\quad \quad \quad = \{q_0, q_1\}$
4. $\quad \quad \quad = [q_0, q_1]$

Similarly,

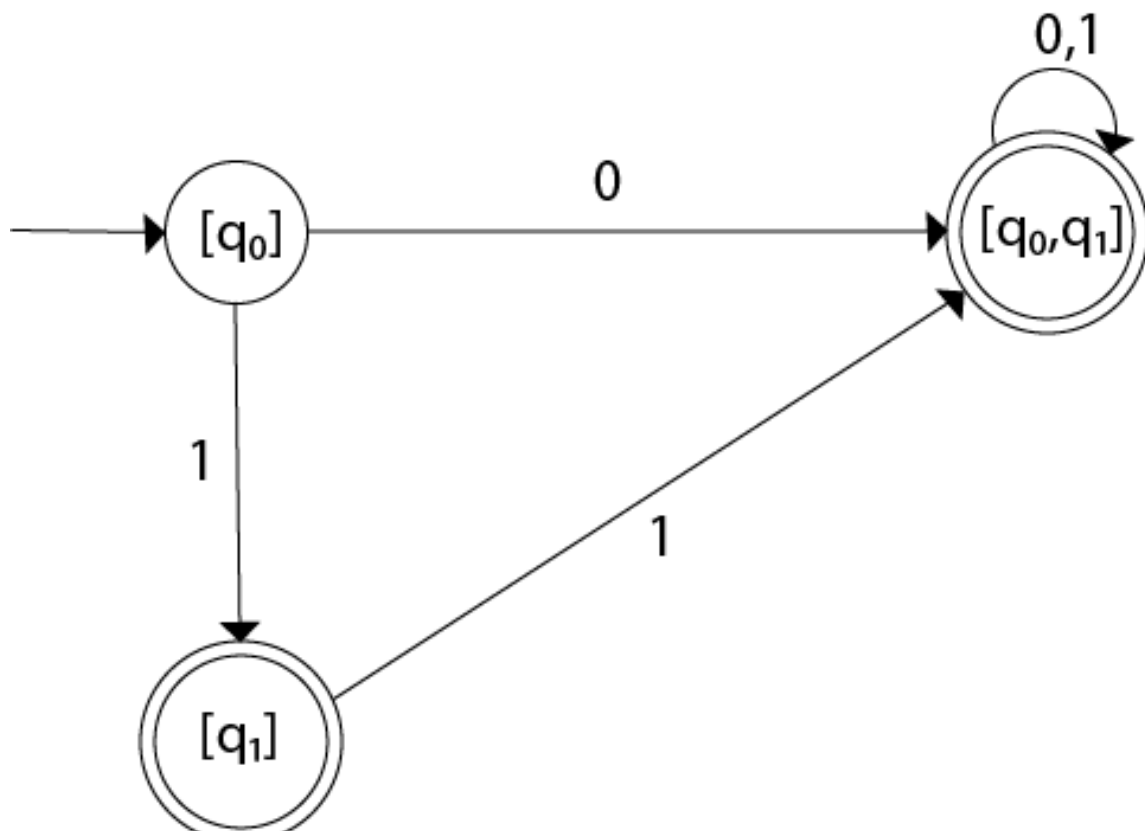
1. $\delta'([q_0, q_1], 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$
2. $\quad \quad \quad = \{q_1\} \cup \{q_0, q_1\}$
3. $\quad \quad \quad = \{q_0, q_1\}$
4. $\quad \quad \quad = [q_0, q_1]$

As in the given NFA, q₁ is a final state, then in DFA wherever, q₁ exists that state becomes a final state. Hence in the DFA, final states are [q₁] and [q₀, q₁]. Therefore set of final states $F = \{[q_1], [q_0, q_1]\}$.

The transition table for the constructed DFA will be:

State	0	1
$\rightarrow[q_0]$	$[q_0, q_1]$	$[q_1]$
$*[q_1]$	ϕ	$[q_0, q_1]$
$*[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_1]$

The Transition diagram will be:

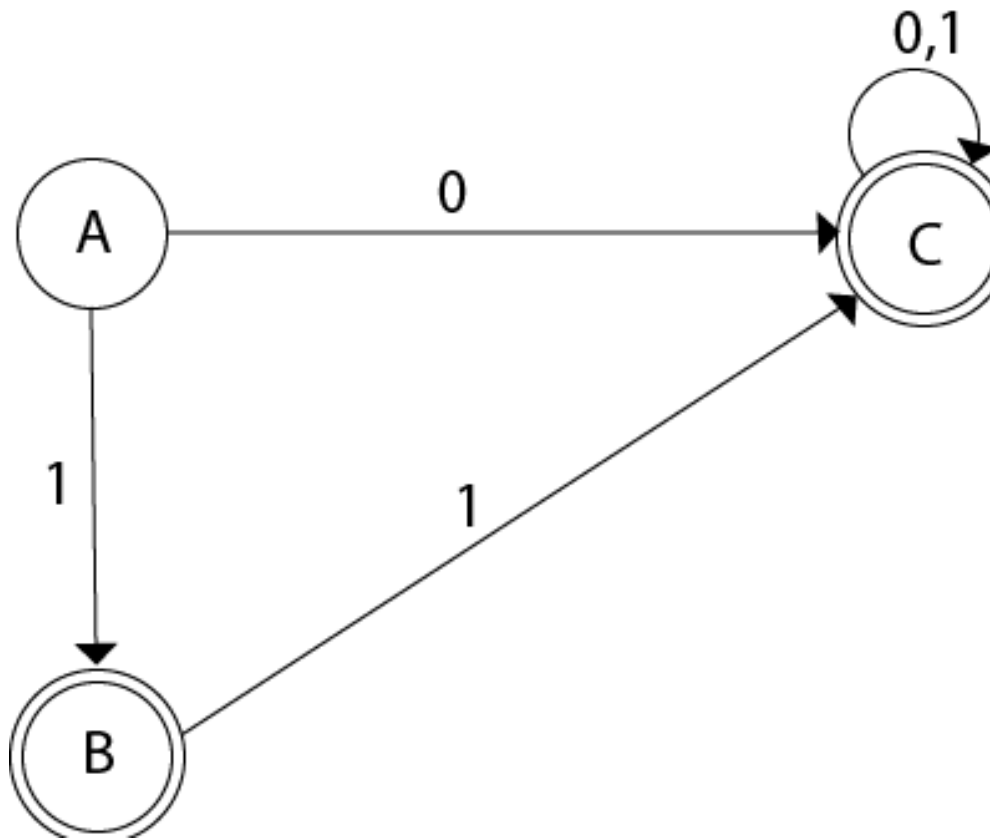


Even we can change the name of the states of DFA.

Suppose

1. A = $[q_0]$
2. B = $[q_1]$
3. C = $[q_0, q_1]$

With these new names the DFA will be as follows:



Optimizing DFA

To optimize the DFA you have to follow the various steps. These are as follows:

Step 1: Remove all the states that are unreachable from the initial state via any set of the transition of DFA.

Step 2: Draw the transition table for all pair of states.

Step 3: Now split the transition table into two tables T1 and T2. T1 contains all final states and T2 contains non-final states.

Step 4: Find the similar rows from T1 such that:

1. $\delta(q, a) = p$
2. $\delta(r, a) = p$

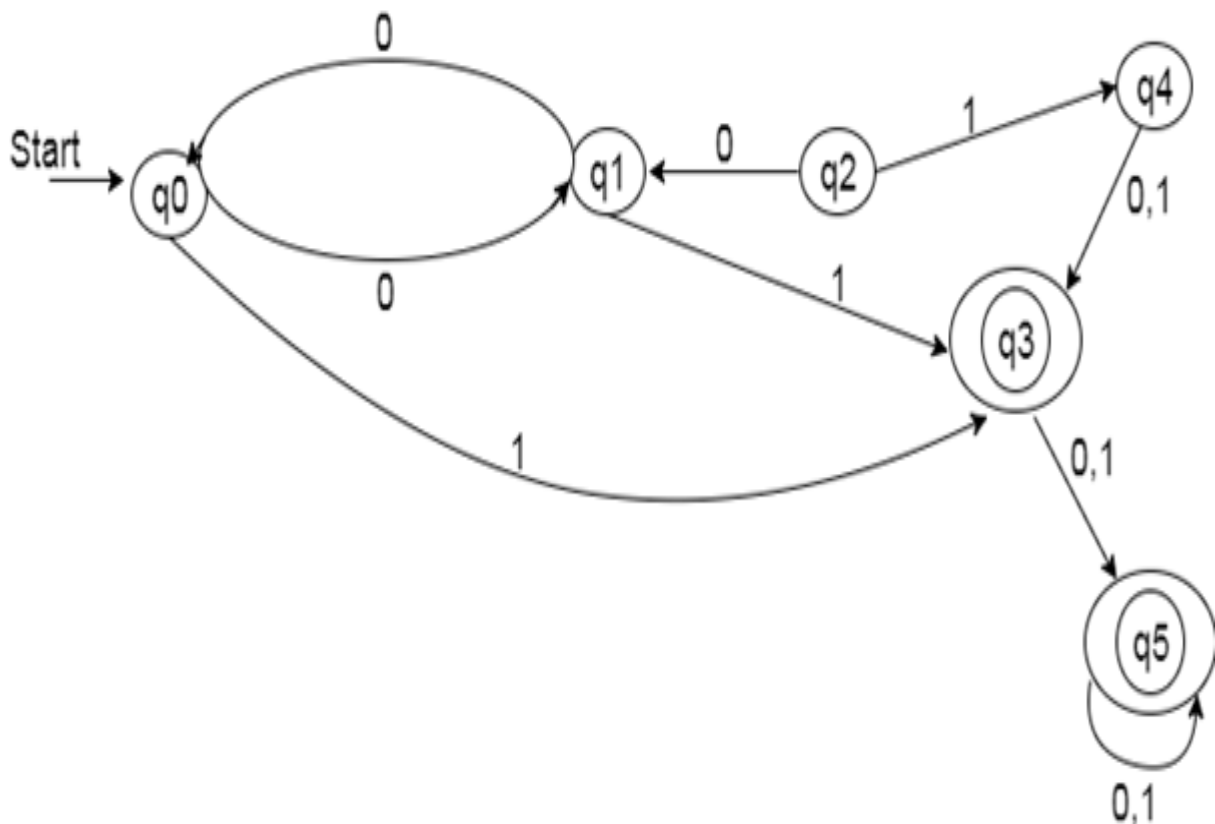
That means, find the two states which have same value of a and b and remove one of them.

Step 5: Repeat step 3 until there is no similar rows are available in the transition table T1.

Step 6: Repeat step 3 and step 4 for table T2 also.

Step 7: Now combine the reduced T1 and T2 tables. The combined transition table is the transition table of minimized DFA.

Example



Solution:

Step 1: In the given DFA, q2 and q4 are the unreachable states so remove them.

Step 2: Draw the transition table for rest of the states.

State	0	1
<p>→ q0</p> <p>q1</p> <p>*q3</p> <p>*q5</p>	<p>q1</p> <p>q0</p> <p>q5</p> <p>q5</p>	<p>q3</p> <p>q3</p> <p>q5</p> <p>q5</p>

Step 3:

Now divide rows of transition table into two sets as:

1. One set contains those rows, which start from non-final states:

State	0	1
<p>q0</p> <p>q1</p>	<p>q1</p> <p>q0</p>	<p>q3</p> <p>q3</p>

2. Other set contains those rows, which starts from final states.

State	0	1
<p>q3</p> <p>q5</p>	<p>q5</p> <p>q5</p>	<p>q5</p> <p>q5</p>

Step 4: Set 1 has no similar rows so set 1 will be the same.

Step 5: In set 2, row 1 and row 2 are similar since q_3 and q_5 transit to same state on 0 and 1. So skip q_5 and then replace q_3 in the rest.

State	0	1
q_3	q_3	q_3

Step 6: Now combine set 1 and set 2 as:

State	0	1
$\rightarrow q_0$ q_1 $*q_3$	q_1 q_0 q_3	q_3 q_3 q_3

Now it is the transition table of minimized DFA.

Transition diagram of minimized DFA:

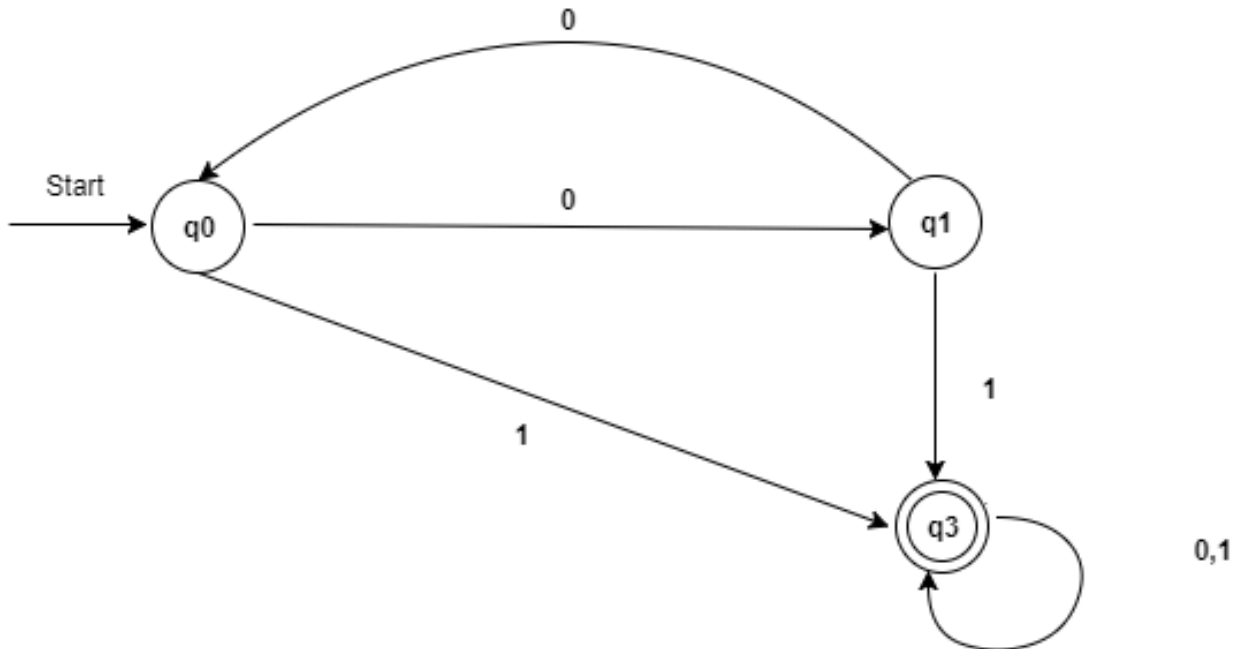


Fig: Minimized DFA

FA with output :-

Moore machine

In the theory of computation, a Moore machine is a finite-state machine whose output values are determined only by its current state. This is in contrast to a Mealy machine, whose (Mealy) output values are determined both by its current state and by the values of its inputs. The Moore machine is named after Edward F. Moore, who presented the concept in a 1956 paper, "Gedanken-experiments on Sequential Machines."

A Moore machine can be defined as a 6-tuple consisting of the following:

- A finite set of states
- A start state (also called initial state) which is an element of
- A finite set called the input alphabet
- A finite set called the output alphabet
- A transition function mapping a state and the input alphabet to the next state
- An output function mapping each state to the output alphabet

A Moore machine can be regarded as a restricted type of finite-state transducer.

Mealy machine

Finite automata may have outputs corresponding to each transition. There are two types of finite state machines that generate output –

- Mealy Machine
- Moore machine

Mealy Machine

A Mealy Machine is an FSM whose output depends on the present state as well as the present input.

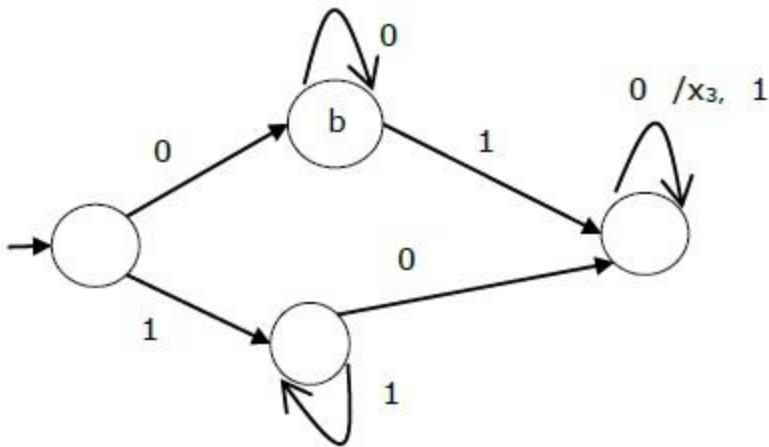
It can be described by a 6 tuple $(Q, \Sigma, O, \delta, X, q_0)$ where –

- Q is a finite set of states.
- Σ is a finite set of symbols called the input alphabet.
- O is a finite set of symbols called the output alphabet.
- δ is the input transition function where $\delta: Q \times \Sigma \rightarrow Q$
- X is the output transition function where $X: Q \times \Sigma \rightarrow O$
- q_0 is the initial state from where any input is processed ($q_0 \in Q$).

The state table of a Mealy Machine is shown below –

Present state	Next state			
	input = 0		input = 1	
	State	Output	State	Output
→ a	b	x_1	c	x_1
b	b	x_2	d	x_3
c	d	x_3	c	x_1
d	d	x_3	d	x_2

The state diagram of the above Mealy Machine is –



Moore Machine

Moore machine is an FSM whose outputs depend on only the present state.

A Moore machine can be described by a 6 tuple $(Q, \Sigma, O, \delta, X, q_0)$ where –

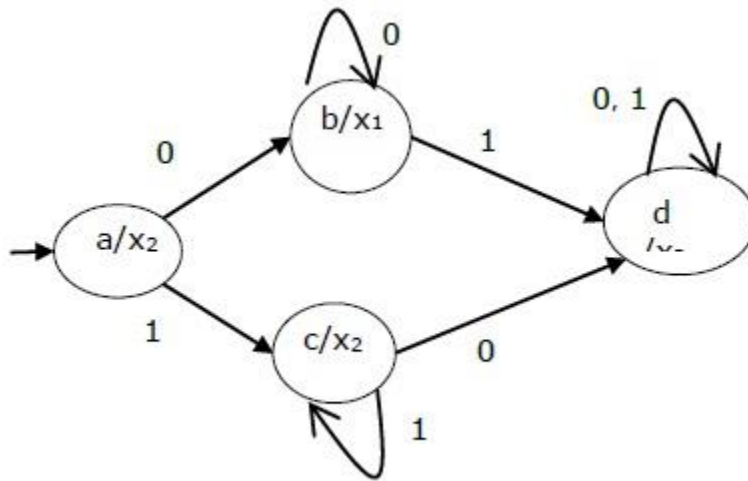
- Q is a finite set of states.
- Σ is a finite set of symbols called the input alphabet.
- O is a finite set of symbols called the output alphabet.
- δ is the input transition function where $\delta: Q \times \Sigma \rightarrow Q$
- X is the output transition function where $X: Q \rightarrow O$
- q_0 is the initial state from where any input is processed ($q_0 \in Q$).

The state table of a Moore Machine is shown below –

Present state	Next State		Output
	Input = 0	Input = 1	
→ a	b	c	x_2
b	b	d	x_1

c	c	d	x_2
d	d	d	x_3

The state diagram of the above Moore Machine is –



Mealy Machine vs. Moore Machine

The following table highlights the points that differentiate a Mealy Machine from a Moore Machine.

Mealy Machine	Moore Machine
Output depends both upon the present state and the present input	Output depends only upon the present state.
Generally, it has fewer states than Moore Machine.	Generally, it has more states than Mealy Machine.
The value of the output function is a function of the transitions and the changes, when the input logic on the present state is done.	The value of the output function is a function of the current state and the changes at the clock edges, whenever state changes occur.

Mealy machines react faster to inputs. They generally react in the same clock cycle.

In Moore machines, more logic is required to decode the outputs resulting in more circuit delays. They generally react one clock cycle later.

Moore Machine to Mealy Machine

Algorithm 4

Input – Moore Machine

Output – Mealy Machine

Step 1 – Take a blank Mealy Machine transition table format.

Step 2 – Copy all the Moore Machine transition states into this table format.

Step 3 – Check the present states and their corresponding outputs in the Moore Machine state table; if for a state Q_i output is m , copy it into the output columns of the Mealy Machine state table wherever Q_i appears in the next state.

Example

Let us consider the following Moore machine –

Present State	Next State		Output
	a = 0	a = 1	
→ a	d	b	1
b	a	d	0
c	c	c	0
d	b	a	1

Now we apply Algorithm 4 to convert it to Mealy Machine.

Step 1 & 2 –

Present State	Next State			
	a = 0		a = 1	
	State	Output	State	Output
→ a	d		b	
b	a		d	
c	c		c	
d	b		a	

Step 3 –

Present State	Next State			
	a = 0		a = 1	
	State	Output	State	Output
=> a	d	1	b	0
b	a	1	d	1
c	c	0	c	0
d	b	0	a	1

Mealy Machine to Moore Machine

Algorithm 5

Input – Mealy Machine

Output – Moore Machine

Step 1 – Calculate the number of different outputs for each state (Q_i) that are available in the state table of the Mealy machine.

Step 2 – If all the outputs of Q_i are same, copy state Q_i . If it has n distinct outputs, break Q_i into n states as Q_{in} where $n = 0, 1, 2, \dots$

Step 3 – If the output of the initial state is 1, insert a new initial state at the beginning which gives 0 output.

Example

Let us consider the following Mealy Machine –

Present State	Next State			
	a = 0		a = 1	
	Next State	Output	Next State	Output
→ a	d	0	b	1
b	a	1	d	0
c	c	1	c	0
d	b	0	a	1

Here, states 'a' and 'd' give only 1 and 0 outputs respectively, so we retain states 'a' and 'd'. But states 'b' and 'c' produce different outputs (1 and 0). So, we divide **b** into **b₀**, **b₁** and **c** into **c₀**, **c₁**.

Present State	Next State	Output
---------------	------------	--------

	a = 0	a = 1	
$\rightarrow a$	d	b_1	1
b_0	a	d	0
b_1	a	d	1
c_0	c_1	C_0	0
c_1	c_1	C_0	1
d	b_0	a	0

UNIT-V

Non-regular language:

Pumping Lemma

Let L be a regular language. Then there exists a constant ' c ' such that for every string w in L –

$$|w| \geq c$$

We can break w into three strings, $w = xyz$, such that –

- $|y| > 0$
- $|xy| \leq c$
- For all $k \geq 0$, the string xy^kz is also in L .

Applications of Pumping Lemma

Pumping Lemma is to be applied to show that certain languages are not regular. It should never be used to show a language is regular.

- If L is regular, it satisfies Pumping Lemma.
- If L does not satisfy Pumping Lemma, it is non-regular.

Method to prove that a language L is not regular

- At first, we have to assume that L is regular.
- So, the pumping lemma should hold for L .
- Use the pumping lemma to obtain a contradiction –
 - Select w such that $|w| \geq c$
 - Select y such that $|y| \geq 1$
 - Select x such that $|xy| \leq c$
 - Assign the remaining string to z .
 - Select k such that the resulting string is not in L .

Hence L is not regular.

Problem

Prove that $L = \{a^i b^i \mid i \geq 0\}$ is not regular.

Solution –

- At first, we assume that L is regular and n is the number of states.
- Let $w = a^n b^n$. Thus $|w| = 2n \geq n$.
- By pumping lemma, let $w = xyz$, where $|xy| \leq n$.
- Let $x = a^p$, $y = a^q$, and $z = a^r b^n$, where $p + q + r = n$, $p \neq 0$, $q \neq 0$, $r \neq 0$. Thus $|y| \neq 0$.
- Let $k = 2$. Then $xy^2z = a^p a^{2q} a^r b^n$.
- Number of a 's = $(p + 2q + r) = (p + q + r) + q = n + q$
- Hence, $xy^2z = a^{n+q} b^n$. Since $q \neq 0$, xy^2z is not of the form $a^n b^n$.
- Thus, xy^2z is not in L . Hence L is not regular.

Pushdown Automata

- Pushdown automata is a way to implement a CFG in the same way we design DFA for a regular grammar. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information.
- Pushdown automata is simply an NFA augmented with an "external stack memory". The addition of stack is used to provide a last-in-first-out memory management capability to Pushdown automata. Pushdown automata can store an unbounded amount of information on the stack. It can access a limited amount of information on the stack. A PDA can push an element onto the top of the stack and pop off an element from the top of the stack. To read an element into the stack, the top elements must be popped off and are lost.
- A PDA is more powerful than FA. Any language which can be acceptable by FA can also be acceptable by PDA. PDA also accepts a class of language which even cannot be accepted by FA. Thus PDA is much more superior to FA.

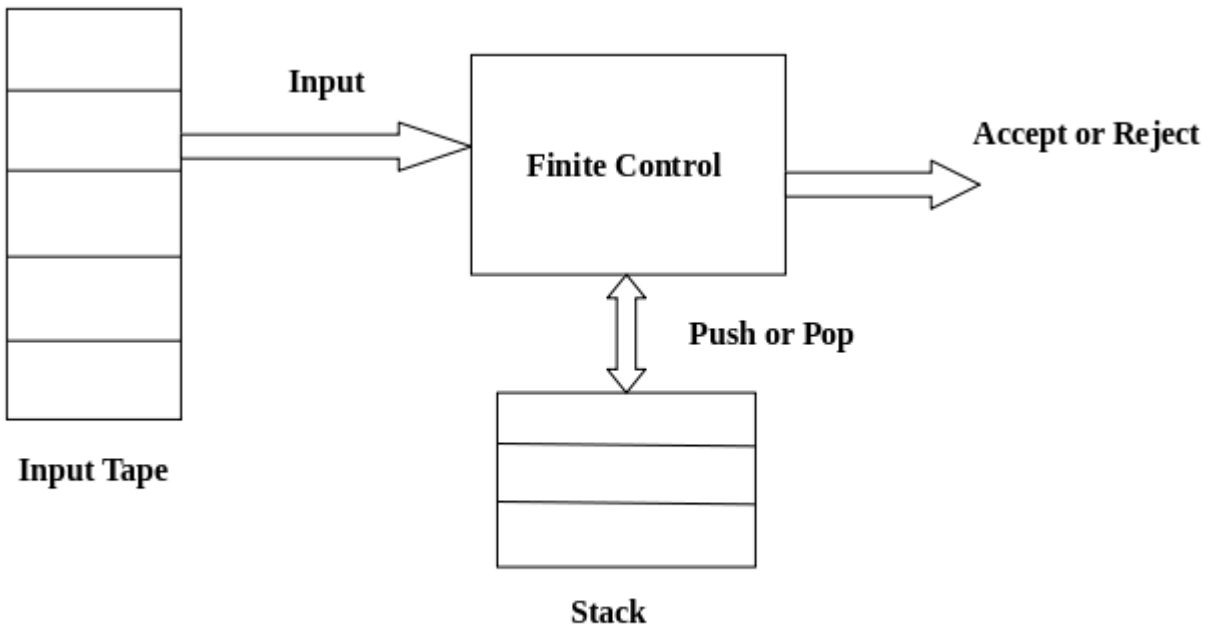


Fig: Pushdown Automata

PDA Components:

Input tape: The input tape is divided in many cells or symbols. The input head is read-only and may only move from left to right, one symbol at a time.

Finite control: The finite control has some pointer which points the current symbol which is to be read.

Stack: The stack is a structure in which we can push and remove the items from one end only. It has an infinite size. In PDA, the stack is used to store the items temporarily.

Formal definition of PDA:

The PDA can be defined as a collection of 7 components:

Q: the finite set of states

Σ : the input set

Γ : a stack symbol which can be pushed and popped from the stack

q0: the initial state

Z: a start symbol which is in Γ .

F: a set of final states

δ : mapping function which is used for moving from current state to next state.

Instantaneous Description (ID)

ID is an informal notation of how a PDA computes an input string and make a decision that string is accepted or rejected.

An instantaneous description is a triple (q, w, α) where:

q describes the current state.

w describes the remaining input.

α describes the stack contents, top at the left.

Turnstile Notation:

\vdash sign describes the turnstile notation and represents one move.

\vdash^* sign describes a sequence of moves.

For example,

$(p, b, T) \vdash (q, w, \alpha)$

In the above example, while taking a transition from state p to q, the input symbol 'b' is consumed, and the top of the stack 'T' is represented by a new string α .

Example 1:

Design a PDA for accepting a language $\{a^n b^{2n} \mid n \geq 1\}$.

Solution: In this language, n number of a's should be followed by 2n number of b's. Hence, we will apply a very simple logic, and that is if we read single 'a', we will push two a's onto the stack. As soon as we read 'b' then for every single 'b' only one 'a' should get popped from the stack.

The ID can be constructed as follows:

1. $\delta(q_0, a, Z) = (q_0, aaZ)$
2. $\delta(q_0, a, a) = (q_0, aaa)$

Now when we read b, we will change the state from q0 to q1 and start popping corresponding 'a'. Hence,

$$1. \delta(q_0, b, a) = (q_1, \epsilon)$$

Thus this process of popping 'b' will be repeated unless all the symbols are read. Note that popping action occurs in state q1 only.

$$1. \delta(q_1, b, a) = (q_1, \epsilon)$$

After reading all b's, all the corresponding a's should get popped. Hence when we read ϵ as input symbol then there should be nothing in the stack. Hence the move will be:

$$1. \delta(q_1, \epsilon, Z) = (q_2, \epsilon)$$

Where

$$\text{PDA} = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, Z\}, \delta, q_0, Z, \{q_2\})$$

We can summarize the ID as:

1. $\delta(q_0, a, Z) = (q_0, aaZ)$
2. $\delta(q_0, a, a) = (q_0, aaa)$
3. $\delta(q_0, b, a) = (q_1, \epsilon)$
4. $\delta(q_1, b, a) = (q_1, \epsilon)$
5. $\delta(q_1, \epsilon, Z) = (q_2, \epsilon)$

Now we will simulate this PDA for the input string "aaabbbbb".

1. $\delta(q_0, aaabbbbb, Z) \vdash \delta(q_0, aabbbbb, aaZ)$
2. $\vdash \delta(q_0, abbbbb, aaaaZ)$
3. $\vdash \delta(q_0, bbbbb, aaaaaZ)$
4. $\vdash \delta(q_1, bbbbb, aaaaaZ)$
5. $\vdash \delta(q_1, bbbb, aaaaZ)$
6. $\vdash \delta(q_1, bbb, aaaZ)$
7. $\vdash \delta(q_1, bb, aaZ)$
8. $\vdash \delta(q_1, b, aZ)$
9. $\vdash \delta(q_1, \epsilon, Z)$
10. $\vdash \delta(q_2, \epsilon)$
11. ACCEPT

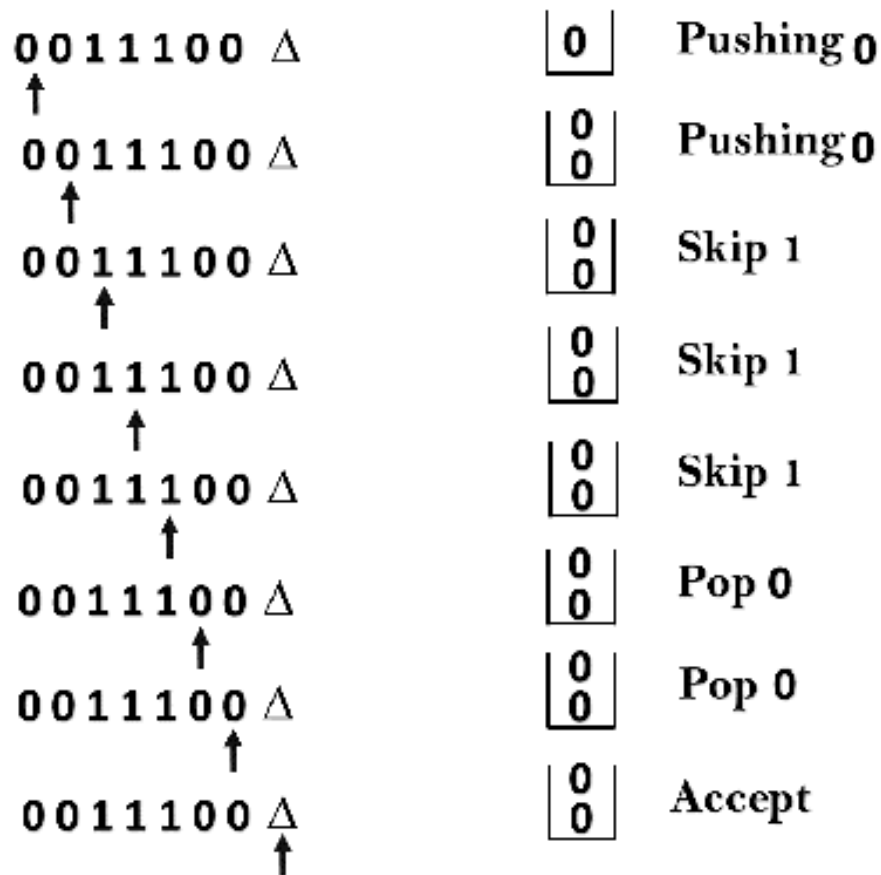
Example 2:

Design a PDA for accepting a language $\{0^n 1^m 0^n \mid m, n \geq 1\}$.

Solution: In this PDA, n number of 0's are followed by any number of 1's followed n number of 0's. Hence the logic for design of such PDA will be as follows:

Push all 0's onto the stack on encountering first 0's. Then if we read 1, just do nothing. Then read 0, and on each read of 0, pop one 0 from the stack.

For instance:



This scenario can be written in the ID form as:

1. $\delta(q_0, 0, Z) = \delta(q_0, 0Z)$
2. $\delta(q_0, 0, 0) = \delta(q_0, 00)$
3. $\delta(q_0, 1, 0) = \delta(q_1, 0)$
4. $\delta(q_0, 1, 0) = \delta(q_1, 0)$
5. $\delta(q_1, 0, 0) = \delta(q_1, \epsilon)$
6. $\delta(q_0, \epsilon, Z) = \delta(q_2, Z)$ (ACCEPT state)

Now we will simulate this PDA for the input string "0011100".

1. $\delta(q_0, 0011100, Z) \vdash \delta(q_0, 011100, 0Z)$

2. $\vdash \delta(q_0, 11100, 00Z)$
3. $\vdash \delta(q_0, 1100, 00Z)$
4. $\vdash \delta(q_1, 100, 00Z)$
5. $\vdash \delta(q_1, 00, 00Z)$
6. $\vdash \delta(q_1, 0, 0Z)$
7. $\vdash \delta(q_1, \epsilon, Z)$
8. $\vdash \delta(q_2, Z)$
9. ACCEPT

Introduction to Turing Machine and its elementary applications to recognition of a language and computation of functions

A Turing Machine is an accepting device which accepts the languages (recursively enumerable set) generated by type 0 grammars. It was invented in 1936 by Alan Turing.

Definition

A Turing Machine (TM) is a mathematical model which consists of an infinite length tape divided into cells on which input is given. It consists of a head which reads the input tape. A state register stores the state of the Turing machine. After reading an input symbol, it is replaced with another symbol, its internal state is changed, and it moves from one cell to the right or left. If the TM reaches the final state, the input string is accepted, otherwise rejected.

A TM can be formally described as a 7-tuple $(Q, X, \Sigma, \delta, q_0, B, F)$ where –

- **Q** is a finite set of states
- **X** is the tape alphabet
- Σ is the input alphabet
- δ is a transition function; $\delta : Q \times X \rightarrow Q \times X \times \{\text{Left_shift}, \text{Right_shift}\}$.
- q_0 is the initial state
- **B** is the blank symbol
- **F** is the set of final states

Comparison with the previous automaton

The following table shows a comparison of how a Turing machine differs from Finite Automaton and Pushdown Automaton.

Machine	Stack Data Structure	Deterministic?
Finite Automaton	N.A	Yes
Pushdown Automaton	Last In First Out(LIFO)	No
Turing Machine	Infinite tape	Yes

Example of Turing machine

Turing machine $M = (Q, X, \Sigma, \delta, q_0, B, F)$ with

- $Q = \{q_0, q_1, q_2, q_f\}$
- $X = \{a, b\}$
- $\Sigma = \{1\}$
- $q_0 = \{q_0\}$
- $B = \text{blank symbol}$
- $F = \{q_f\}$

δ is given by –

Tape alphabet symbol	Present State ' q_0 '	Present State ' q_1 '	Present State ' q_2 '
a	1R q_1	1L q_0	1L q_f
b	1L q_2	1R q_1	1R q_f

Here the transition 1R q_1 implies that the write symbol is 1, the tape moves right, and the next state is q_1 . Similarly, the transition 1L q_2 implies that the write symbol is 1, the tape moves left, and the next state is q_2 .

Time and Space Complexity of a Turing Machine

For a Turing machine, the time complexity refers to the measure of the number of times the tape moves when the machine is initialized for some input symbols and the space complexity is the number of cells of the tape written.

Time complexity all reasonable functions –

$$T(n) = O(n \log n)$$

TM's space complexity –

$$S(n) = O(n)$$

A TM accepts a language if it enters into a final state for any input string w . A language is recursively enumerable (generated by Type-0 grammar) if it is accepted by a Turing machine.

A TM decides a language if it accepts it and enters into a rejecting state for any input not in the language. A language is recursive if it is decided by a Turing machine.

There may be some cases where a TM does not stop. Such TM accepts the language, but it does not decide it.

Designing a Turing Machine

The basic guidelines of designing a Turing machine have been explained below with the help of a couple of examples.

Example 1

Design a TM to recognize all strings consisting of an odd number of α 's.

Solution

The Turing machine **M** can be constructed by the following moves –

- Let q_1 be the initial state.
- If **M** is in q_1 ; on scanning α , it enters the state q_2 and writes **B** (blank).
- If **M** is in q_2 ; on scanning α , it enters the state q_1 and writes **B** (blank).
- From the above moves, we can see that **M** enters the state q_1 if it scans an even number of α 's, and it enters the state q_2 if it scans an odd number of α 's. Hence q_2 is the only accepting state.

Hence,

$$M = \{\{q_1, q_2\}, \{1\}, \{1, B\}, \delta, q_1, B, \{q_2\}\}$$

where δ is given by –

Tape alphabet symbol	Present State ' q_1 '	Present State ' q_2 '
α	BR q_2	BR q_1

Example 2

Design a Turing Machine that reads a string representing a binary number and erases all leading 0's in the string. However, if the string comprises of only 0's, it keeps one 0.

Solution

Let us assume that the input string is terminated by a blank symbol, B, at each end of the string.

The Turing Machine, **M**, can be constructed by the following moves –

- Let q_0 be the initial state.
- If **M** is in q_0 , on reading 0, it moves right, enters the state q_1 and erases 0. On reading 1, it enters the state q_2 and moves right.
- If **M** is in q_1 , on reading 0, it moves right and erases 0, i.e., it replaces 0's by B's. On reaching the leftmost 1, it enters q_2 and moves right. If it reaches B, i.e., the string comprises of only 0's, it moves left and enters the state q_3 .
- If **M** is in q_2 , on reading either 0 or 1, it moves right. On reaching B, it moves left and enters the state q_4 . This validates that the string comprises only of 0's and 1's.
- If **M** is in q_3 , it replaces B by 0, moves left and reaches the final state q_f .
- If **M** is in q_4 , on reading either 0 or 1, it moves left. On reaching the beginning of the string, i.e., when it reads B, it reaches the final state q_f .

Hence,

$$M = \{\{q_0, q_1, q_2, q_3, q_4, q_f\}, \{0, 1, B\}, \{1, B\}, \delta, q_0, B, \{q_f\}\}$$

where δ is given by –

Tape alphabet symbol	Present State ' q_0 '	Present State ' q_1 '	Present State ' q_2 '	Present State ' q_3 '	Present State ' q_4 '
0	BR q_1	BR q_1	OR q_2	-	OL q_4
1	1R q_2	1R q_2	1R q_2	-	1L q_4
B	BR q_1	BL q_3	BL q_4	OL q_f	BR q_f