UGC NET Paper 2
Computer Science and Applications
Code No.:(87)

Unit 1: Discrete Structures and Optimization

1. Mathematical Logic
2. Sets and Relations
3. Counting, Mathematical Induction and Discrete Probability
4. Group Theory
5. Graph Theory
6. Boolean Algebra
7. Optimization

Mathematical Logic:
1. Propositional and Predicate Logic
2. Propositional Equivalences
3. Normal Forms
4. Predicates and Quantifiers
5. Nested Quantifiers
6. Rules of Inference

Sets and Relations:
1. Set Operations
2. Representation and Properties of Relations
3. Equivalence Relations
4. Partially Ordering.

Counting, Mathematical Induction and Discrete Probability:
1. Basics of Counting,
2. Pigeonhole Principle
3. Permutations and Combinations
4. Inclusion- Exclusion Principle
5. Mathematical Induction
6. Probability
7. Bayes' Theorem

Group Theory:
1. Groups, Subgroups
2. Semi Groups
3. Product and Quotients of Algebraic Structures
4. Isomorphism
5. Homomorphism
6. Automorphism
7. Rings
8. Integral Domains
9. Fields
10. Applications of Group Theory

Graph Theory:
1. Simple Graph
2. Multigraph
3. Weighted Graph
4. Paths and Circuits
5. Shortest Paths in Weighted Graphs
6. Eulerian Paths and Circuits
7. Hamiltonian Paths and Circuits
8. Planner graph
9. Graph Coloring
10. Bipartite Graphs
11. Trees and Rooted Trees
12. Prefix Codes
13. Tree Traversals
14. Spanning Trees and Cut-Sets

Boolean Algebra:
1. Boolean Functions and its Representation
2. Simplifications of Boolean Functions.

Optimization:
1. Linear Programming - Mathematical Model
2. Graphical Solution
3. Simplex and Dual Simplex Method
4. Sensitive Analysis
5. Integer Programming
6. Transportation and Assignment Models
7. PERT-CPM: Diagram Representation
8. Critical Path Calculations
9. Resource Levelling
10. Cost Consideration in Project Scheduling.

1. Mathematical Logic

1. Propositional and Predicate Logic
2. Propositional Equivalences
3. Normal Forms
4. Predicates and Quantifiers
5. Nested Quantifiers
6. Rules of Inference

<u>Propositional Logic / Boolean Logic:</u> A proposition is a collection of declarative statements that has either a truth value "true" or a truth value "false".

Examples of Propositions

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

Propositional Logic consists of:

1. propositional variables
2. Connectives: connectives connect the propositional variables.
   - ❑ OR (∨)
   - ❑ AND (∧)
   - ❑ Negation/ NOT (¬)
   - ❑ Implication / if-then / conditional statement (→)
   - ❑ If and only if / bi-conditional statement (⇔).

Terminologies:

1. Tautologies: always true
2. Contradictions: always false
3. Contingency: has both some true and some false values

OR (∨)		
A	**B**	**A ∨ B**
True	True	True
True	False	True
False	True	True
False	False	False

AND (∧)		
A	**B**	**A ∧ B**
True	True	True
True	False	False
False	True	False
False	False	False

Negation NOT (¬)	
A	**¬ A**
True	False
False	True

Implication if-then (→)		
A	**B**	**A → B**
True	True	True
True	False	False
False	True	True
False	False	True

bi-conditional If and only if (⇔)		
A	**B**	**A ⇔ B**
True	True	True
True	False	False
False	True	False
False	False	True

Note:

- Inverse, Converse, and Contra-positive

Implication / if-then / conditional statement (→) has two parts −

p→q where,

1. Hypothesis, p
2. Conclusion, q

Inverse, Converse, and Contra-positive of conditional statement:

Eg: p→q || IF you do your homework -> THEN you will not be punished

- Inverse: ¬p→¬q || IF you do not do your homework, THEN you will be punished
- Converse: q→p || IF you will not be punished, THEN you do your homework
- Contra-positive: ¬q→¬p || IF you are punished, THEN you did not do your homework

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⇔Y is a tautology.

Testing by 1st method (Matching truth table)						
A	**B**	**A ∨ B**	**¬ (A ∨ B)**	**¬ A**	**¬ B**	**[(¬ A) ∧ (¬ 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**
truth values of ¬(A∨B)and[(¬A)∧(¬B)] are same, hence the statements are equivalent.						

Example:

Prove ¬(A∨B)and[(¬A)∧(¬B)] are equivalent

Tautologies: always true		
Exp.1	*Exp.2*	*Tautologies*
True	*True*	*True*
True	*False*	*True*
False	*True*	*True*
False	*False*	*True*

bi-conditional If and only if (⇔)		
Exp.1	*Exp.2*	*Exp.1 ⇔ Exp.2*
True	*True*	*True*
True	*False*	*False*
False	*True*	*False*
False	*False*	*True*

Testing by 2nd method (Bi-conditionality)				
A	**B**	**¬ (A ∨ B)**	**[(¬ A) ∧ (¬ B)]**	[¬ (A ∨ B)] ⇔ [(¬ A) ∧ (¬ B)]
True	True	**False**	**False**	True
True	False	**False**	**False**	True
False	True	**False**	**False**	True
False	False	**True**	**True**	True
As [¬(A∨B)]⇔[(¬A)∧(¬B)] is a tautology, the statements are equivalent.				

Normal Forms:

We can convert any proposition in two normal forms −

1. 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
   - $(A \lor B) \land (A \lor C) \land (B \lor C \lor D)$
   - $(P \cup Q) \cap (Q \cup R)$

2. 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
   - $(A \land B) \lor (A \land C) \lor (B \land C \land D)$
   - $(P \cap Q) \cup (Q \cap R)$

Predicate Logic:

- A predicate is an expression of one or more variables defined on some specific domain.
- A predicate with variables can be made a proposition by either assigning a value to the variable or by quantifying the variable.
- Predicate Logic deals with predicates, which are propositions containing variables.
- examples of predicates:
  - Let E(x, y) denote "x = y"
  - Let X(a, b, c) denote "a + b + c = 0"
  - Let M(x, y) denote "x is married to y”

Quantifiers: There are two types of quantifier s in predicate logic −

Domain *specifies possible values of the variable under consideration.*

1. **Universal Quantifier ∀:** *denoted by the symbol ∀ (while the symbol ∀ is read as "For all")*

- Notation: ∀xP(x) *{ Read as: "For all x, P(x)", "For every x, P(x)" }*
- Universal quantifier of P(x) is the statement " P(x) is true for all values of x in the domain"
- Example − Let P(x) = "x+1 > x" where domain = set of positive integers
- Check: x=1 ; 1+1 > 1 ∴ P(x) is true
    - x=2 ; 2+1 > 2 ∴ P(x) is true
    - x=3 ; 3+1 > 3 ∴ P(x) is true
    - Since P(x) is true for all values of x ∴ P(x) = "x+1 > x" is a Universal Quantifier

2. **Existential Quantifier ∃:** *denoted by the symbol ∃ (while the symbol ∃ is read as "There exists")*

- Notation: ∃xP(x) *{ Read as: "There exist an x such that P(x)", "There is at least one x such that P(x)", "For some x P(x)" }*
- Existential quantifier of P(x) is the statement " There exists an element x in the domain such that P(x)"
- Example − Let P(x) = "$x^2 > 10$" where domain = {1,2,3,4}
- Check: x=1 ; $1^2 > 10$ ∴ P(x) is false
    - x=2 ; $2^2 > 10$ ∴ P(x) is false
    - x=3 ; $3^2 > 10$ ∴ P(x) is false
    - x=4 ; $4^2 > 10$ ∴ P(x) is true
    - Since P(x) is true for x=4 ∴ P(x) = "$x^2 > 10$" is an Existential Quantifier

Nested Quantifiers: Two quantifiers are nested if one is within the scope of the other.

Different combinations of nested quantifiers

- $\forall x \forall y\ Q(x,y)$ *Here Order of quantifiers doesn't matter* *i.e.,* $\forall x \forall y\ Q(x,y) = \forall y \forall x\ Q(x,y)$
- $\forall x \exists y\ Q(x,y)$ *Here Order of quantifiers does matter* *i.e.,* $\forall x \exists y\ Q(x,y) \neq \exists y \forall x\ Q(x,y)$
- $\exists x \forall y\ Q(x,y)$ *Here Order of quantifiers does matter* *i.e.,* $\exists x \forall y\ Q(x,y) \neq \forall y \exists x\ Q(x,y)$
- $\exists x \exists y\ Q(x,y)$ *Here Order of quantifiers doesn't matter* *i.e.,* $\exists x \exists y\ Q(x,y) = \exists y \exists x\ Q(x,y)$

Note:

Anything within the scope of a quantifier can be thought as a prepositional function

For example: $\forall x \exists y\ Q(x,y)$

Here essential quantifier is within the scope of a universal quantifier

Therefore, we can write it as: $\forall x\ P(x)$

$\forall x \exists y\ Q(x,y) = \forall x\ P(x)$ where, $P(x) = \exists y\ Q(x,y)$

Rules of Inference:

- Rule of inference are templates for constructing valid arguments.
- Inference means driving a conclusion from evidences.
- Evidences are premises or hypothesis.
- Rule of inference are themselves arguments that are used to construct valid arguments
- The symbol "∴ ", (read as therefore) is placed before the conclusion.

	Name	Rule of Inference	Example:
1	**Addition**	*P* ∴P∨Q	P= He studies very hard P∨Q = Either he studies very hard Or he is a very bad student Q = he is a very bad student
2	**Conjunction**	*P* *Q* ∴P∧Q	P = He studies very hard Q = He is the best boy in the class P∧Q = He studies very hard and he is the best boy in the class
3	**Simplification**	P∧Q ∴P	P∧Q = He studies very hard and he is the best boy in the class P = He studies very hard
4	**Modus Ponens**	*P -> Q* *P* ∴Q	P→Q = If you have a password, then you can log on to facebook P = You have a password Q = You can log on to facebook
5	**Modus Tollens**	*P -> Q* ¬*Q* ∴¬P	P→Q = If you have a password, then you can log on to facebook ¬Q = You cannot log on to facebook ¬P = You do not have a password
6	**Disjunctive Syllogism**	*P* ∨*Q* ¬*P* ∴Q	¬P = The ice cream is not vanilla flavored P∨Q = The ice cream is either vanilla flavored or chocolate flavored Q = The ice cream is chocolate flavored
7	**Hypothetical Syllogism**	*P -> Q* *Q* → *R* ∴P → *R*	P→Q = If it rains, I shall not go to school Q→R = If I don't go to school, I won't need to do homework P→R = If it rains, I won't need to do homework
8	**Constructive Dilemma**	*(P -> Q)* ∧ *(R -> S)* P∨R ∴Q∨S	(P→Q) = If it rains, I will take a leave (R→S) = If it is hot outside, I will go for a shower P∨R = Either it will rain or it is hot outside Q∨S = I will take a leave or I will go for a shower
9	**Destructive Dilemma**	*(P -> Q)* ∧ *(R -> S)* ¬Q∨¬S ∴¬P∨¬R	(P→Q) = If it rains, I will take a leave (R→S) = If it is hot outside, I will go for a shower ¬Q∨¬S = Either I will not take a leave or I will not go for a shower ¬P∨¬R = Either it does not rain or it is not hot outside

1. Mathematical Logic

1. Propositional Logic
2. Propositional Equivalences
3. Normal Forms
4. Predicate Logic
5. Quantifiers
6. Nested Quantifiers
7. Rules of Inference