Discrete Mathematics 🔢

Dr. Aamir Alaud Din

Topic 1: Introduction and Foundations

Semester: Spring 2026

💡 Learning Outcomes

By the end of this course you will be able to:

- Develop clear reasoning needed in programming and algorithm design.
- Translate real-world problems into logical form which are the basis of computing.
- Match logical structures to if–else, loops, and boolean expressions.
- Validate logic used in software, hardware circuits, and AI.
- Bridge theory with practice.

🤔 The Why Section

A) Programming correctness

- Every program is a series of logical decisions.

Example: Authentication

```
if (user_entered_ID == stored_ID && password_is_correct)
    allow_login;
```

- This is exactly **p ∧ q** logic.

🤔 The Why Section

- If a programmer misuses logic, a system may:
  - Let unauthorized users enter
  - Crash on wrong input
  - Reject valid actions

🤔 The Why Section

B) Algorithm design and proofs

- Before implementing an algorithm, we reason about:
  - Correctness
  - Termination
  - Output validity
- Logical proof methods guarantee:
  - The algorithm works for ALL cases, not just the ones we test.

🤔 The Why Section

C) Computer hardware and circuits

- Truth tables map directly to logic gates:

p	q	p ∧ q
0	0	0
0	1	0
1	0	0
1	1	1

- This is **AND** gate in CPU design.

🤔 The Why Section

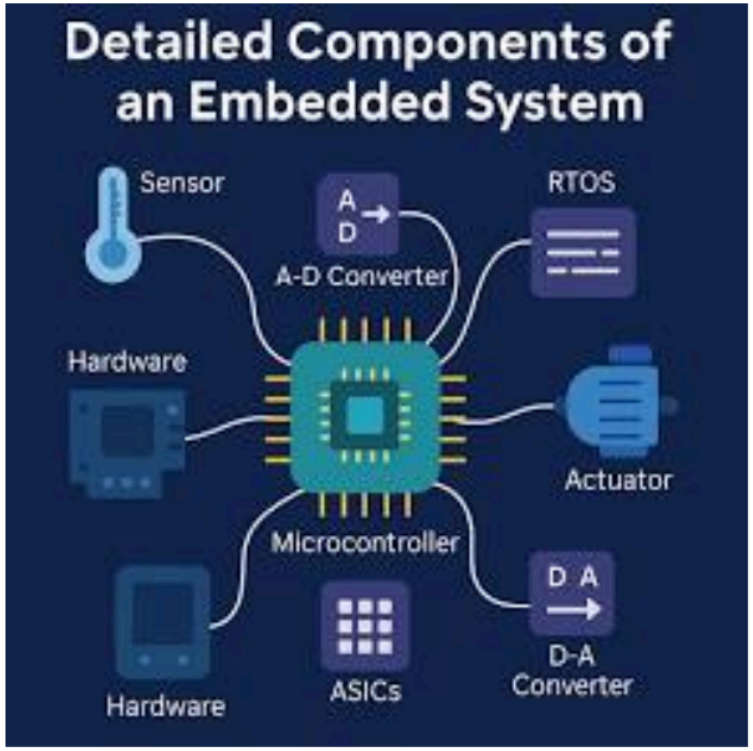
- Without truth tables, it is impossible to design:
  - CPUs
  - Microcontrollers
  - Embedded systems

Embedded Systems

- Embedded systems are specialized computer systems with dedicated functions, built as components within larger mechanical or electronic devices, like the microcontroller in a microwave or the system in a car's anti-lock brakes etc.

🤔 The Why Section

Embedded Systems: Visual Understanding



🤔 The Why Section

D) Networking and internet protocols

- Network routing uses logical rules:

```
if packet_is_valid ∧ destination_exists → forward
else drop
```

- Routing decisions = logical implication

🤔 The Why Section

E) Cybersecurity and encryption

- Security policies are logic-based:

```
If user is admin → allow access
```

```
If key mismatch → deny
```

- Firewalls use boolean rule sets
- Cryptography relies on logic + discrete structures, not calculus.

🤔 The Why Section

F) Artificial Intelligence

- AI uses:
  - Logical inference
  - Decision trees (truth conditions)
  - Constraint satisfaction problems
- Even machine learning models execute decisions based on logic.

🤔 The Why Section

⚠️ What Happens If We Don’t Study These?

Incorrect software behavior

- Misunderstanding AND/OR leads to logical bugs:

```
if(user_is_admin || password_correct)
→ Massive security hole
```

(should be **AND**, not **OR**)

🤔 The Why Section

⚠️ What Happens If We Don’t Study These?

Insecure systems

- If implications are misunderstood:

```
p → q ≠ q → p
```

- Programmer may accidentally give access backward.

🤔 The Why Section

⚠️ What Happens If We Don’t Study These?

Hardware design failure

- Without truth tables, designing gates & circuits becomes guesswork.

Broken algorithms and programs

- Students will memorize syntax without understanding meaning.

Poor debugging ability

- Debugging the program becomes challenging.

Discrete Mathematics

Definition

Discrete mathematics is the branch of mathematics that deals with objects that are separate, distinct, and countable, rather than continuous.

- **Discrete** means not smooth, not continuous
- Opposite of calculus, which studies smooth curves and continuous change
- Discrete math studies structures that can be listed, counted, or separated one by one.

Discrete Mathematics

Key Idea

- If you can count the objects (even if millions), it is discrete.
- If you need calculus to describe it, it is continuous.
- Different objects with three 0's and 1's ($2^3 = 8$)

```
000                     001                     010
100                     110                     101
011                     111
```

Discrete Mathematics

Why is this important?

- Nearly all of Computer Science is based on discrete objects, because computers:
  - Use 0 and 1
  - Store finite data
  - Execute step-by-step instructions
  - Manipulate symbols, not real physical quantities
- So discrete math provides the mathematical foundation for CS.

Discrete Mathematics

What Topics Does Discrete Math Include?

- Here are the major areas:

Topic	What It Studies	CS Example
Logic	True/false statements	Designing `if-else` and verifying algorithms
Sets & Functions	Collections and mappings	Databases, hashing
Algorithms	Step-by-step procedures	Sorting, searching

Discrete Mathematics

Topic	What It Studies	CS Example
Relations	How objects are connected	Social networks, dependency graphs
Graphs & Trees	Nodes and edges	Internet routing, file systems
Combinatorics	Counting possibilities	Password strength
Number Theory	Integers & modular math	Cryptography
Probability	Randomness and risk	Machine learning basics

Discrete Mathematics

Example 1: Boolean Logic

- A computer program checks:

```
if x > 5 and y < 10:
    print("Valid")
```

- This is based on propositions and logical connectives (AND, OR, NOT), which are discrete math.

Discrete Mathematics

Example 2: Graph Theory in Networks

- People as nodes
- Friendships as edges
- Facebook, WhatsApp, routing on the Internet → all use graph algorithms.

Discrete Mathematics

Example 3: Counting & Combinatorics

- How many 4-digit PIN codes are possible?
- Digits: 0–9 → 10 choices per position
- $Total = 10^4 = 10,000$
- Used in:
  - Password security
  - Brute-force cracking analysis

Discrete Mathematics

Example 4: Modular Arithmetic (Crypto)

- Encrypted messages use formulas like:

$$C \equiv P^e \ (mod \ \ n)$$

$$mod \ \ n = \text{Remainder when } P^e \text{ is divided by } n$$

- Used in:
  - Online banking
  - WhatsApp encryption

Discrete Mathematics

Example 5: Trees in File Systems

- Your folders on Linux/Windows form a tree:

```
(D:\)
├─ Spring 2026
│    └─ Lectures
└─ Admin Jobs
     └─ Coordinator
```

- Tree structures are discrete.

Discrete Mathematics

Applicability

- Development of algorithms
- Development of programs
- Debugging of the programs
- Cyber security of the program usage

Porposition

- A proposition is a declarative sentence (that is, a sentence that declares a fact) that is either true or false, but not both.

Example 1

a) Washigton, D. C. is the capital of USA. $\rightarrow$ True

b) Toronto is the capital of Canada. $\rightarrow$ False

c) $2 + 2 = 4 \rightarrow$ True

d) $1 + 3 = 3 \rightarrow$ False

Proposition

- The following statements are not propositions.

a) What time is it? $\rightarrow$ Question
b) Read this carefully. $\rightarrow$ Instruction
c) $x + 1 = 3 \rightarrow$ Not proved
d) $y - z = 2x \rightarrow$ Not proved

Proposition

- Propositional variables (or sentential variables) are represented by letters *p*, *q*, *r*, and *s etc.*
- It is similar to using variable name for numerical values, for example, $g = 9.8\ m/s^2$ in physics and `g_acc = 9.8` in python programming.
- The truth of proposition is represented by T and represented by F if the proposition is not true or false.
- If a proposition can't be simplified, it is called atomic proposition.
- The area of logic that deals with propositions is called the propositional calculus or propo-
  sitional logic.

Proposition

- We now turn our attention to methods for producing new propositions from those that we already have.
- These methods were discussed by the English mathematician George Boole in 1854 in his book The Laws of Thought.
- Many mathematical statements are constructed by combining one or more propositions.
- New propositions, called compound propositions, are formed from existing propositions using logical operators.

Proposition

Definition: Negation Proposition

Let p be a proposition. The negation of p, denoted by $\neg p$ (also denoted by $\bar{p}$), is the statement
"It is not the case that p."
The proposition $\neg p$ is read "not p." The truth value of the negation of p, $\neg p$, is the opposite of the truth value of $$p$.

Proposition

- The negation of the proposition “Michael’s PC runs Linux” is “It is not the case that Michael’s PC runs Linux”.
- The negation of the proposition “Vandana’s smartphone has at least 32 GB of memory” is “It is not the case that Vandana’s smartphone has at least 32 GB of memory”

Proposition

Truth Table for Negation of a Proposition

Proposition (p)	Negation Proposition ($\neg p$)
T	F
F	T

Proposition

Definition: Conjunction Proposition

Let p and q be propositions. The conjunction of p and q, denoted by $p \wedge q$, is the proposition "p and q". The conjunction $p \wedge q$ is true when both p and q are true and is false otherwise.

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

Definition: Disjunction Proposition

Let p and q be propositions. The disjunction of p and q, denoted by $p \vee q$, is the proposition "p or q." The disjunction $p \vee q$ is false when both p and q are false and is true otherwise.

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

Proposition

Definition: Exclusive or Proposition

Let p and q be propositions. The exclusive or of p and q, denoted by $p \oplus q$ (or $p \ XOR \ q$), is the proposition that is true when exactly one of p and q is true and is false otherwise.

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

🎉 Thanks!

Questions?