📘 Discrete Mathematics — Lecture 2 Notes

Truth Tables

Welcome back!
In Lecture 1, we discovered **how simple statements become powerful when combined with logic**.
Now in Lecture 2, we move one step deeper:

> **Truth Tables — the microscope that shows what a logical statement is *really* doing.**

Truth tables force a computer-scientist mindset:
no vibes, no opinions — just **all possible cases**, laid out cleanly.

Let's break it down like Feynman would explain it to a curious teenager.

⭐ Why Truth Tables Matter

Imagine you're debugging a giant if-else condition in code:

```
if (loggedIn and verified) or not(isBanned):
```

How do you know when the condition is true?
You can't trust intuition — our brains get messy when conditions interact.

Truth tables rescue us.

A truth table says:

> "Let's test *every* possible combination of TRUE and FALSE, and write down what the statement becomes."

That's it.
This simple tool is the backbone of:

- logic gates
- digital circuits
- Boolean algebra
- AI reasoning
- compiler design
- algorithm correctness

Truth tables **turn English logic → machine logic**.

🧠 Truth Tables for Three Important Expressions

We will construct truth tables for:

1. **~p ∧ q**
2. **~p ∧ (q ∨ ~r)**
3. **(p ∨ q) ∧ ~(p ∧ q)** — the famous XOR

Let's go one by one.

---

1) 🔍 Truth Table for ~p ∧ q

This statement reads:

> *"NOT p AND q."*

It is true only when:

- p is false
- q is true

Truth table:

p	**q**	**~p**	**~p ∧ q**
T	T	F	F
T	F	F	F
F	T	T	**T**
F	F	T	F

Only **one** row gives TRUE:
when *p is false* and *q is true*.

---

2) 🔍 Truth Table for ~p ∧ (q ∨ ~r)

This one is slightly trickier.
Break it down like a programmer:

- First compute ~p
- Then compute ~r
- Then compute **(q ∨ ~r)**
- Finally compute **~p ∧ (q ∨ ~r)**

Truth table:

p	**q**	**r**	**~p**	**~r**	**q ∨ ~r**	**~p ∧ (q ∨ ~r)**
T	T	T	F	F	T	F

T	F	T	F	F	F	F
F	T	F	T	T	T	**T**
F	F	T	T	F	F	F
F	T	T	T	F	T	**T**
T	T	F	F	T	T	F
T	F	F	F	T	T	F
F	F	F	T	T	T	**T**

A bit longer, but the pattern becomes clear:
TRUE happens when:

- p is FALSE, and
- q is TRUE **or** r is FALSE

---

3) 🔍 Truth Table for (p ∨ q) ∧ ~(p ∧ q)

This is the **Exclusive OR (XOR)** —
the logic behind:

- odd parity
- bit toggling
- digital adders
- many AI decision circuits

The expression says:

> “p or q — but NOT both.”

Truth table:

p	q	p ∨ q	p ∧ q	~(p ∧ q)	(p ∨ q) ∧ ~(p ∧ q)
T	T	T	T	F	F
T	F	T	F	T	**T**
F	T	T	F	T	**T**
F	F	F	F	T	F

This matches XOR perfectly.

🟧 Inclusive OR vs Exclusive OR

English is sloppy.

When your mother says:

> “Beta, get milk or bread.”

She might mean:

- milk
- bread
- **or both** (inclusive OR)

But if a friend says:

> “Tomorrow I’ll be in Lahore or Islamabad.”

He absolutely doesn’t mean both.
This is **exclusive OR**.

In logic:

- ∨ = **inclusive OR**
- ⊕ = **exclusive OR (XOR)**

Formula for XOR:

$$p \oplus q = (p \vee q) \wedge \sim(p \wedge q)$$

🧩 Logical Equivalence

Two statements are **logically equivalent** if their columns match exactly in a truth table.

Example:

$\sim(\sim p) \equiv p$
Double negation just brings you back.

Truth table verifies this cleanly.

🧨 De Morgan's Laws (The Two Golden Rules)

These are the "hacking tools" of logic.

1. **Negation flips AND → OR**

   $\sim(p \wedge q) \equiv \sim p \vee \sim q$

2. **Negation flips OR → AND**

   $\sim(p \vee q) \equiv \sim p \wedge \sim q$

These laws help you:

- simplify logic

- negate conditions
- rewrite if-else blocks
- build circuit diagrams

🎯 Example (Using De Morgan on Inequalities)

Negate the statement:

$-1 < x \leq 4$

Equivalent logical structure:

- $x > -1$
- $x \leq 4$

Negation:

$x \leq -1$ OR $x > 4$

This is the outside region.

🔥 Tautology & Contradiction

Tautology (Always True)

Example:

$p \vee \sim p$

No matter what p is, this is TRUE.

Contradiction (Always False)

Example:

$p \wedge \sim p$

Impossible for both to happen.

🧱 Laws of Logic (Your Boolean Toolbox)

🧱 1. Commutative Law

Order doesn't matter.

AND:

$p \wedge q \equiv q \wedge p$
"Tea AND Biscuits" is the same as "Biscuits AND Tea."

OR:

$p \vee q \equiv q \vee p$
"Tea OR Coffee" is the same as "Coffee OR Tea."

Why it's true:
You're not changing the choices — only the *order*.

🧱 2. Associative Law

When grouping things, brackets don't matter.

AND:

$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$

Example:
("Wake up AND brush teeth") AND eat breakfast
= Wake up AND ("brush teeth AND eat breakfast")

All three must happen anyway.

OR:

$(p \vee q) \vee r \equiv p \vee (q \vee r)$

Example:
(Milk OR Bread) OR Eggs = Milk OR (Bread OR Eggs)

Why it's true:
You're still combining the same list of conditions.

🧱 3. Distributive Law

AND distributes over OR, and OR distributes over AND.
This is the trickiest, but also the most useful.

AND distributes over OR:

$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$

Example:
"You enter the building IF:

- you have ID
- OR you have a visitor pass."

Rewrite:
"You enter IF:

- ID + visitor list
- OR visitor pass + visitor list."

Makes sense logically.

OR distributes over AND:

$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

Example:
"You can attend the event if:

- you paid
  OR
- (you're a student AND registered)."

Rewrite:
"You can attend if:

- (you paid OR you're a student)
  AND
- (you paid OR you're registered)."

Why it's true:
Think of it like expanding brackets in algebra.

🧱 4. Identity Law

Combining with 'true' or 'false' keeps things unchanged.

t = true
c = false

AND:

$p \wedge t \equiv p$
"Condition AND True" changes nothing.

OR:

$p \vee c \equiv p$
"Condition OR False" changes nothing.

Example:
If you say:
“I'll go out if I'm free AND the sun rises.”
Sun always rises → doesn’t change anything.

🧱 5. Negation Law

These define tautology & contradiction.

Tautology (always true):

$p \vee \sim p \equiv t$
“Either it’s raining or it's not raining.”

Contradiction (always false):

$p \wedge \sim p \equiv c$
“It’s raining AND it’s not raining.”

🧱 6. Double Negation

Cancel the negative.

$\sim(\sim p) \equiv p$

Example:
“Not (not hungry)” = “Hungry.”

The two NOTs kill each other.

🧱 7. Idempotent Law

Repeating the same condition doesn’t change anything.

AND:

$p \wedge p \equiv p$

OR:

$p \vee p \equiv p$

Example:
"Bring milk OR bring milk" is still just "bring milk."

🧱 8. De Morgan's Laws

Negating a group flips AND ↔ OR.

First:

$\sim(p \wedge q) \equiv \sim p \vee \sim q$
Not (p AND q) = (Not p) OR (Not q)

Example:
"Not (both students passed)"
means
"Either student 1 failed OR student 2 failed."

Second:

$\sim(p \vee q) \equiv \sim p \wedge \sim q$
Not (p OR q) = (Not p) AND (Not q)

Example:
"Not (tea OR coffee)"
means
"You want neither — no tea AND no coffee."

🧱 9. Universal Bound

True and False are dominant operations.

OR with True:

$p \vee t \equiv t$
If one option is always true, OR becomes useless — result is always true.

Example:
"I'll go outside if I finish work OR if sun rises."
Sun always rises → outcome always true.

AND with False:

$p \wedge c \equiv c$
If one requirement is impossible, AND collapses.

Example:
"You get the job if you pass interview AND time freezes."
Time will never freeze → condition always false.

📝 Lecture 2 Summary

By now, you can:

✔ Build truth tables
✔ Detect logical equivalence
✔ Understand inclusive vs exclusive OR
✔ Apply De Morgan's laws
✔ Identify tautologies & contradictions
✔ Simplify logical expressions confidently

This is the **mathematical brain** of computer science.